
HARSH KASHYAP

CSE 4

101917088

hkashyap_be19@thapar.edu

A Practical activity Report submitted
for Operating Systems (UCS303)

OPERATING SYSTEM



Computer Science and Engineering
Patiala Campus
2020

Submitted to
Dr. Ashwani Kumar

Assignment 1

Question 1

Introduction to Operating System.

Answer -

It acts as an intermediary between user and hardware. Resource manager which manages system resources in an unbiased fashion both h/w and s/w. It provides a platform on which other applications are installed.

An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers. An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

There are 3 key elements of an operating system, which are: (1) Abstractions (process, thread, file, socket, memory), (2) Mechanisms (create, schedule, open, write, allocate), and (3) Policies (LRU, EDF)

Some of the important functions of an Operating System.

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Some popular Operating Systems include Linux Operating System, Windows Operating System etc.

Question 2

Introduction to Linux/Unix

Answer -

Linux is a community of open-source Unix like operating systems that are based on the Linux Kernel. It is a free and open-source operating system and the source code can be modified and distributed to anyone. Linux was created for personal computers gradually it is used in other machines like servers, mainframe computers, supercomputers, etc. Linux is also used in embedded systems like routers, automation controls, televisions, digital video recorders, video game consoles, smartwatches, etc.

Linux is one of the popular versions of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX with minor differences. Source Code is available to the general public in case of Linux only. Linux is portable and can be executed on various hard drives but Unix isn't portable. Unix contains the Command Line Interface whereas Linux contains the Command Line as well as Graphical User Interface. Examples of Unix are Solaris, HP UNIX, BSD, etc. whereas examples of Linux are Ubuntu, Fedora, Red Hat, etc.

Following are some of the important features of Linux Operating System.

- Portable
 - Open Source
 - Multi-User Multiprogramming
 - Hierarchical File System
 - Shell
 - Security
-

Question 3

Architecture of Linux/Unix

Answer -

The basic architecture of Linux/Unix is :

- Kernel:** Kernel is the core of the Linux based operating system. It virtualizes the common hardware resources of the computer to provide each process with its virtual resources. This makes the process seem as it is the sole process running on the machine. The kernel is also responsible for preventing and mitigating conflicts between different processes.
 - System Library:** It is the special types of functions that are used to implement the functionality of the operating system.
 - Shell:** It is an interface to the kernel which hides the complexity of the kernel's functions from the users. It takes commands from the user and executes the kernel's functions.
 - Hardware Layer:** This layer consists all peripheral devices like RAM/ HDD/ CPU etc.
 - System Utility:** It provides the functionalities of an operating system to the user.
-

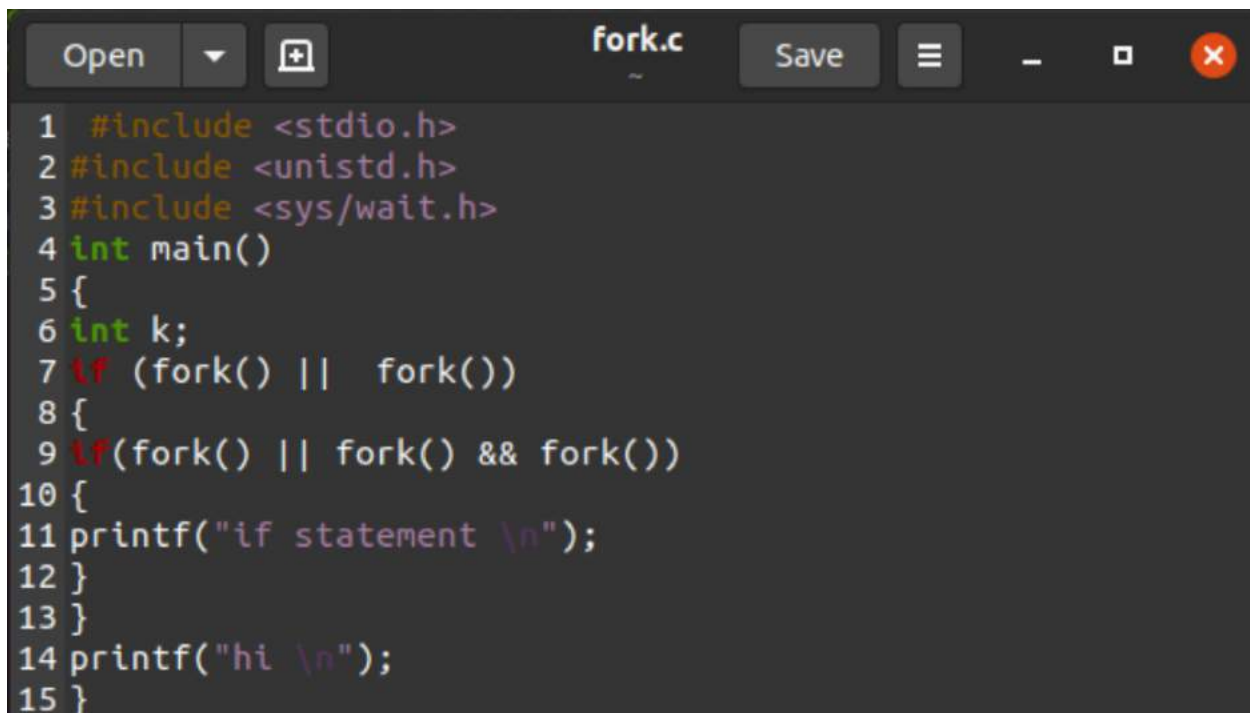
Question 4

System Call execution.

Answer -

A system call is a mechanism that provides the interface between a process and the operating system. It is a programmatic method in which a computer program requests a service from the kernel of the OS. System call offers the services of the operating system to the user programs via API (Application Programming Interface). System calls are the only entry points for the kernel system. A program which calls open, fork, read, write (and many others) makes a system call. System calls are how a program enters the kernel to perform some task. Programs use system calls to perform a variety of operations such as: creating processes, doing network and file IO, and much more.

When a program in user mode requires access to RAM or a hardware resource, it must ask the kernel to provide access to that resource. This is done via something called a system call. When a program makes a system call, the mode is switched from user mode to kernel mode. This is called a context switch. Then the kernel provides the resource which the program requested. After that, another context switch happens which results in change of mode from kernel mode back to user mode.



```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/wait.h>
4 int main()
5 {
6     int k;
7     if (fork() || fork())
8     {
9         if(fork() || fork() && fork())
10        {
11            printf("if statement \n");
12        }
13    }
14    printf("hi \n");
15 }
```

```

harsh@harsh-VirtualBox:~$ ls
a.out      Downloads  fork.c     hello      Music      Practical  Templates
cricket.txt f1         Games     hello.c    Pictures   Public     Thapar
Desktop    File1      games.c    is         prac       sys        Videos
Documents  fork      games.txt  ispositive prac.c     sys.c

harsh@harsh-VirtualBox:~$ gedit fork.c
harsh@harsh-VirtualBox:~$ gcc -o fork fork.c
harsh@harsh-VirtualBox:~$ ./fork
if statement
hi
harsh@harsh-VirtualBox:~$ if statement
hi
if statement
hi
if statement
hi
hi
hi
hi
hi
hi

```

Question 5

Command structure.

Answer -

A system call is a mechanism that provides the interface between a process and the operating system. It is a programmatic method in which a computer program requests a service from the kernel of the OS. System. A command is a program that tells the Linux system to do something.

To give a command to a UNIX system we type the name of the command along with any associated information, such as filename, and press the key. The typed line is called the command line and UNIX uses a special program, called the shell or command line interpreter, to interpret what you have typed into what you want to do. It has the form:

command [options] [arguments],

where an argument indicates on what the command is to perform its action, usually a file or series of files. An option modifies the command, changing the way it performs.

The basic operation we can perform on command structure are: mkdir, rmdir, echo, rm, find, ls, date, cal, cat etc

Assignment 2

Question 1

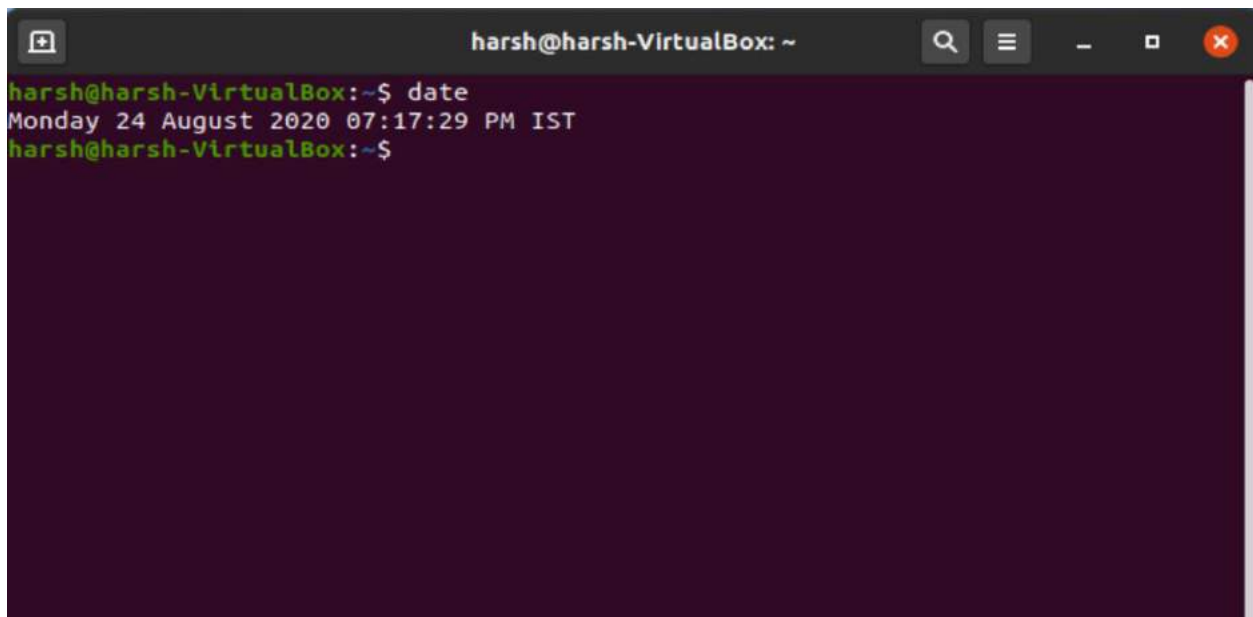
Concept of Shell.

Answer -

A Shell provides you with an interface to the Unix system. It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output.

The shell is a program that takes commands from the keyboard and gives them to the operating system to perform. The prompt, \$, which is called the command prompt, is issued by the shell. While the prompt is displayed, you can type a command. A shell is a program that provides an interface between a user and an operating system (OS) kernel. An OS starts a shell for each user when the user logs in or opens a terminal or console window. A kernel is a program that controls all computer operations. By interfacing with a kernel, a shell provides a way for a user to execute utilities and programs.

Shell reads your input after you press Enter. It determines the command you want to be executed by looking at the first word of your input.

A screenshot of a terminal window titled "harsh@harsh-VirtualBox: ~". The terminal has a dark background with light-colored text. The prompt "harsh@harsh-VirtualBox:~\$" is shown. The user has entered the command "date", and the output "Monday 24 August 2020 07:17:29 PM IST" is displayed. The prompt "harsh@harsh-VirtualBox:~\$" is shown again, indicating the command has been executed.

```
harsh@harsh-VirtualBox:~$ date
Monday 24 August 2020 07:17:29 PM IST
harsh@harsh-VirtualBox:~$
```

Question 2

Types of Shell.

Answer -

The different Shell, their features and their default prompts are also described.

1. The Bourne Shell :

- The Bourne shell is the preferred shell for shell programming because of its compactness and speed.
- For the Bourne shell, the command full-path name is `/bin/sh` and `/sbin/sh`.
- Non-root user default prompt is `$`.
- Root user default prompt is `#`.

2. The C Shell:

- It's a UNIX enhancement.
- Includes convenient programming features, such as built-in arithmetic and C-like expression syntax.
- For the C shell, the command full-path name is `/bin/csh`.
- Non-root user default prompt is `hostname %`.
- Root user default prompt is `hostname #`.

3. The Korn Shell The Korn shell (ksh):

- Supports everything in the Bourne shell and has interactive features comparable to those in the C shell.
- Includes convenient programming features like built-in arithmetic and C-like arrays, functions, and string-manipulation facilities.
- It is faster than the C shell.
- Runs scripts written for the Bourne shell
- Command full-path name is `/bin/ksh`.
- Non-root user default prompt is `$`. Root user default prompt is `#`.

4. The GNU Bourne-Again Shell:

- It is compatible with the Bourne shell and incorporates useful features from the Korn and C shells.
- Has arrow keys that are automatically mapped for command recall and editing.
- Command full-path name is `/bin/bash`.
- The default prompt for a non-root user is `bash-x.xx$`.

5. TC Shell (Denoted as 'tcsh') -

- TC shell is an expansion upon the C shell.
- It has all the same features, but adds the ability to use keystrokes from the Emacs word processor program to edit text on the command line.

Question 3

ls command and its options

Answer -

The ls command supports the following options:

ls - list all files

ls -a: list all files including hidden files. These are files that start with ".".

ls -A: list all files including hidden files except for "." and ".." – these refer to the entries for the current directory, and for the parent directory.

ls -R: list all files recursively, descending down the directory tree from the given path.

ls -l: list the files in long format i.e. with an index number, owner name, group name, size, and permissions.

ls -o: list the files in long format but without the group name.

ls -g: list the files in long format but without the owner name.

ls -i: list the files along with their index number.

ls -s: list the files along with their size.

ls -t: sort the list by time of modification, with the newest at the top.

ls -S: sort the list by size, with the largest at the top.

ls -r: reverse the sorting order.

ls -lt - lists the file in order of time modified.

ls -ltR - lists the file in reverse order of time modified.

ls -lS - lists the file in order of space used.

ls -lsR - lists the file in reverse order of space used.

ls -F - lists the file according to the format of directory and file.

ls -l --block-size=M - lists all the files in Megabyte(can be replaced by K for Kilobyte, G for gigabyte, etc)

```

harsh@harsh-VirtualBox: ~
harsh@harsh-VirtualBox:~$ ls -R
.:
a.out      Downloads  fork.c     hello      Music      Practical  Templates
cricket.txt f1         Games      hello.c    Pictures   Public     Thapar
Desktop    File1      games.c    is         prac       sys        Videos
Documents  fork       games.txt  ispositive prac.c     sys.c

./Desktop:

./Documents:

./Downloads:
alex-mustaros-KiXSWo3cEAI-unsplash.jpg  denys-nevozhai-7f0U7gCulR4-unsplash.jpg
brooke-lark-YSA1IRkGAsg-unsplash.jpg

./Music:

./Pictures:
Thapar

./Pictures/Thapar:
Games1 Games12 Games122 hello.c

./Pictures/Thapar/Games1:

./Pictures/Thapar/Games12:

harsh@harsh-VirtualBox: ~
harsh@harsh-VirtualBox:~$ ls
a.out      Downloads  fork.c     hello      Music      Practical  Templates
cricket.txt f1         Games      hello.c    Pictures   Public     Thapar
Desktop    File1      games.c    is         prac       sys        Videos
Documents  fork       games.txt  ispositive prac.c     sys.c
harsh@harsh-VirtualBox:~$ ls -a
.          cricket.txt games.c     Music      sys
..         Desktop    games.txt  Pictures   sys.c
a.out      Documents  .gnupg     prac       Templates
.bash_history Downloads  hello      prac.c     Thapar
.bash_logout f1         hello.c    Practical  Videos
.bashrc      File1      is         .profile
.byobu       fork       ispositive Public
.cache       fork.c     .local     .ssh
.config      Games      .mozilla   .sudo_as_admin_successful
harsh@harsh-VirtualBox:~$ ls -A
a.out      Desktop    games.c     .mozilla   .ssh
.bash_history Documents  games.txt  Music      .sudo_as_admin_successful
.bash_logout Downloads  .gnupg     Pictures   sys
.bashrc      f1         hello      prac       sys.c
.byobu       File1      hello.c    prac.c     Templates
.cache       fork       is         Practical  Thapar
.config      fork.c     ispositive .profile   Videos
cricket.txt Games      .local     Public

```

```

harsh@harsh-VirtualBox:~$ ls -l
total 180
-rwxrwxr-x 1 harsh harsh 16776 Aug 18 14:17 a.out
-rw-rw-r-- 1 harsh harsh 33 Aug 24 15:25 cricket.txt
drwxr-xr-x 2 harsh harsh 4096 Aug 6 11:33 Desktop
drwxr-xr-x 2 harsh harsh 4096 Aug 2 21:39 Documents
drwxr-xr-x 2 harsh harsh 4096 Aug 23 01:50 Downloads
-rwxrwxrwx 1 harsh root 13 Aug 20 09:11 f1
-rw-rw-r-- 1 harsh harsh 0 Aug 23 04:02 File1
-rwxrwxr-x 1 harsh harsh 16728 Aug 24 16:40 fork
-rw-rw-r-- 1 harsh harsh 191 Aug 16 12:35 fork.c
-rw-rw-r-- 1 harsh harsh 57 Aug 6 09:11 Games
-rw-rw-r-- 1 harsh harsh 27 Aug 14 10:33 games.c
-rw-rw-r-- 1 harsh harsh 29 Aug 24 15:39 games.txt
-rwxrwxr-x 1 harsh harsh 16728 Aug 14 15:38 hello
-rw-rw-r-- 1 harsh harsh 137 Aug 18 14:40 hello.c
-rw-rw-r-- 1 harsh harsh 0 Aug 20 09:32 is
-rw-rw-r-- 1 harsh harsh 7 Aug 20 09:33 ispositive
drwxr-xr-x 2 harsh harsh 4096 Aug 2 21:39 Music
drwxr-xr-x 3 harsh harsh 4096 Aug 23 03:37 Pictures
-rwxrwxr-x 1 harsh harsh 16768 Aug 11 12:49 prac
-rw-rw-r-- 1 harsh harsh 221 Aug 11 12:49 prac.c
drwxrwxrwx 3 harsh harsh 4096 Aug 13 12:55 Practical
drwxr-xr-x 2 harsh harsh 4096 Aug 2 21:39 Public
-rwxrwxr-x 1 harsh harsh 16856 Aug 18 14:41 sys
-rw-rw-r-- 1 harsh harsh 265 Aug 18 14:40 sys.c
drwxr-xr-x 2 harsh harsh 4096 Aug 2 21:39 Templates
drwxrwxr-x 5 harsh harsh 4096 Aug 24 15:34 Thapar
drwxr-xr-x 2 harsh harsh 4096 Aug 2 21:39 Videos

harsh@harsh-VirtualBox:~$ ls -o
total 180
-rwxrwxr-x 1 harsh 16776 Aug 18 14:17 a.out
-rw-rw-r-- 1 harsh 33 Aug 24 15:25 cricket.txt
drwxr-xr-x 2 harsh 4096 Aug 6 11:33 Desktop
drwxr-xr-x 2 harsh 4096 Aug 2 21:39 Documents
drwxr-xr-x 2 harsh 4096 Aug 23 01:50 Downloads
-rwxrwxrwx 1 harsh 13 Aug 20 09:11 f1
-rw-rw-r-- 1 harsh 0 Aug 23 04:02 File1
-rwxrwxr-x 1 harsh 16728 Aug 24 16:40 fork
-rw-rw-r-- 1 harsh 191 Aug 16 12:35 fork.c
-rw-rw-r-- 1 harsh 57 Aug 6 09:11 Games
-rw-rw-r-- 1 harsh 27 Aug 14 10:33 games.c
-rw-rw-r-- 1 harsh 29 Aug 24 15:39 games.txt
-rwxrwxr-x 1 harsh 16728 Aug 14 15:38 hello
-rw-rw-r-- 1 harsh 137 Aug 18 14:40 hello.c
-rw-rw-r-- 1 harsh 0 Aug 20 09:32 is
-rw-rw-r-- 1 harsh 7 Aug 20 09:33 ispositive
drwxr-xr-x 2 harsh 4096 Aug 2 21:39 Music
drwxr-xr-x 3 harsh 4096 Aug 23 03:37 Pictures
-rwxrwxr-x 1 harsh 16768 Aug 11 12:49 prac
-rw-rw-r-- 1 harsh 221 Aug 11 12:49 prac.c
drwxrwxrwx 3 harsh 4096 Aug 13 12:55 Practical
drwxr-xr-x 2 harsh 4096 Aug 2 21:39 Public
-rwxrwxr-x 1 harsh 16856 Aug 18 14:41 sys
-rw-rw-r-- 1 harsh 265 Aug 18 14:40 sys.c
drwxr-xr-x 2 harsh 4096 Aug 2 21:39 Templates
drwxrwxr-x 5 harsh 4096 Aug 24 15:34 Thapar
drwxr-xr-x 2 harsh 4096 Aug 2 21:39 Videos

```



```

harsh@harsh-VirtualBox:~$ ls -l --block-size=K
total 180K
-rwxrwxr-x 1 harsh harsh 17K Aug 18 14:17 a.out
-rw-rw-r-- 1 harsh harsh 1K Aug 24 15:25 cricket.txt
drwxr-xr-x 2 harsh harsh 4K Aug 6 11:33 Desktop
drwxr-xr-x 2 harsh harsh 4K Aug 2 21:39 Documents
drwxr-xr-x 2 harsh harsh 4K Aug 23 01:50 Downloads
-rwxrwxrwx 1 harsh root 1K Aug 20 09:11 f1
-rw-rw-r-- 1 harsh harsh 0K Aug 23 04:02 File1
-rwxrwxr-x 1 harsh harsh 17K Aug 24 16:40 fork
-rw-rw-r-- 1 harsh harsh 1K Aug 16 12:35 fork.c
-rw-rw-r-- 1 harsh harsh 1K Aug 6 09:11 Games
-rw-rw-r-- 1 harsh harsh 1K Aug 14 10:33 games.c
-rw-rw-r-- 1 harsh harsh 1K Aug 24 15:39 games.txt
-rwxrwxr-x 1 harsh harsh 17K Aug 14 15:38 hello
-rw-rw-r-- 1 harsh harsh 1K Aug 18 14:40 hello.c
-rw-rw-r-- 1 harsh harsh 0K Aug 20 09:32 is
-rw-rw-r-- 1 harsh harsh 1K Aug 20 09:33 ispositive
drwxr-xr-x 2 harsh harsh 4K Aug 2 21:39 Music
drwxr-xr-x 3 harsh harsh 4K Aug 23 03:37 Pictures
-rwxrwxr-x 1 harsh harsh 17K Aug 11 12:49 prac
-rw-rw-r-- 1 harsh harsh 1K Aug 11 12:49 prac.c
drwxrwxrwx 3 harsh harsh 4K Aug 13 12:55 Practical
drwxr-xr-x 2 harsh harsh 4K Aug 2 21:39 Public
-rwxrwxr-x 1 harsh harsh 17K Aug 18 14:41 sys
-rw-rw-r-- 1 harsh harsh 1K Aug 18 14:40 sys.c
drwxr-xr-x 2 harsh harsh 4K Aug 2 21:39 Templates
drwxrwxr-x 5 harsh harsh 4K Aug 24 15:34 Thapar
drwxr-xr-x 2 harsh harsh 4K Aug 2 21:39 Videos

```

```

harsh@harsh-VirtualBox:~$ ls -s
total 180
20 a.out      4 f1      4 games.c    4 ispositive  4 Practical  4 Thapar
 4 cricket.txt 0 File1    4 games.txt  4 Music      4 Public     4 Videos
 4 Desktop    20 fork   20 hello     4 Pictures   20 sys
 4 Documents  4 fork.c  4 hello.c   20 prac      4 sys.c
 4 Downloads  4 Games   0 is        4 prac.c     4 Templates

harsh@harsh-VirtualBox:~$ ls -r
Videos      sys      prac      is      games.c  File1     Desktop
Thapar      Public   Pictures  hello.c  Games    f1        cricket.txt
Templates   Practical Music    hello   fork.c   Downloads a.out
sys.c       prac.c   ispositive games.txt fork     Documents

harsh@harsh-VirtualBox:~$ ls -lt
total 180
-rwxrwxr-x 1 harsh harsh 16728 Aug 24 16:40 fork
-rw-rw-r-- 1 harsh harsh 29 Aug 24 15:39 games.txt
drwxrwxr-x 5 harsh harsh 4096 Aug 24 15:34 Thapar
-rw-rw-r-- 1 harsh harsh 33 Aug 24 15:25 cricket.txt
-rw-rw-r-- 1 harsh harsh 0 Aug 23 04:02 File1
drwxr-xr-x 3 harsh harsh 4096 Aug 23 03:37 Pictures
drwxr-xr-x 2 harsh harsh 4096 Aug 23 01:50 Downloads
-rw-rw-r-- 1 harsh harsh 7 Aug 20 09:33 ispositive
-rw-rw-r-- 1 harsh harsh 0 Aug 20 09:32 is
-rwxrwxrwx 1 harsh root 13 Aug 20 09:11 f1
-rwxrwxr-x 1 harsh harsh 16856 Aug 18 14:41 sys
-rw-rw-r-- 1 harsh harsh 265 Aug 18 14:40 sys.c
-rw-rw-r-- 1 harsh harsh 137 Aug 18 14:40 hello.c

```

Question 4

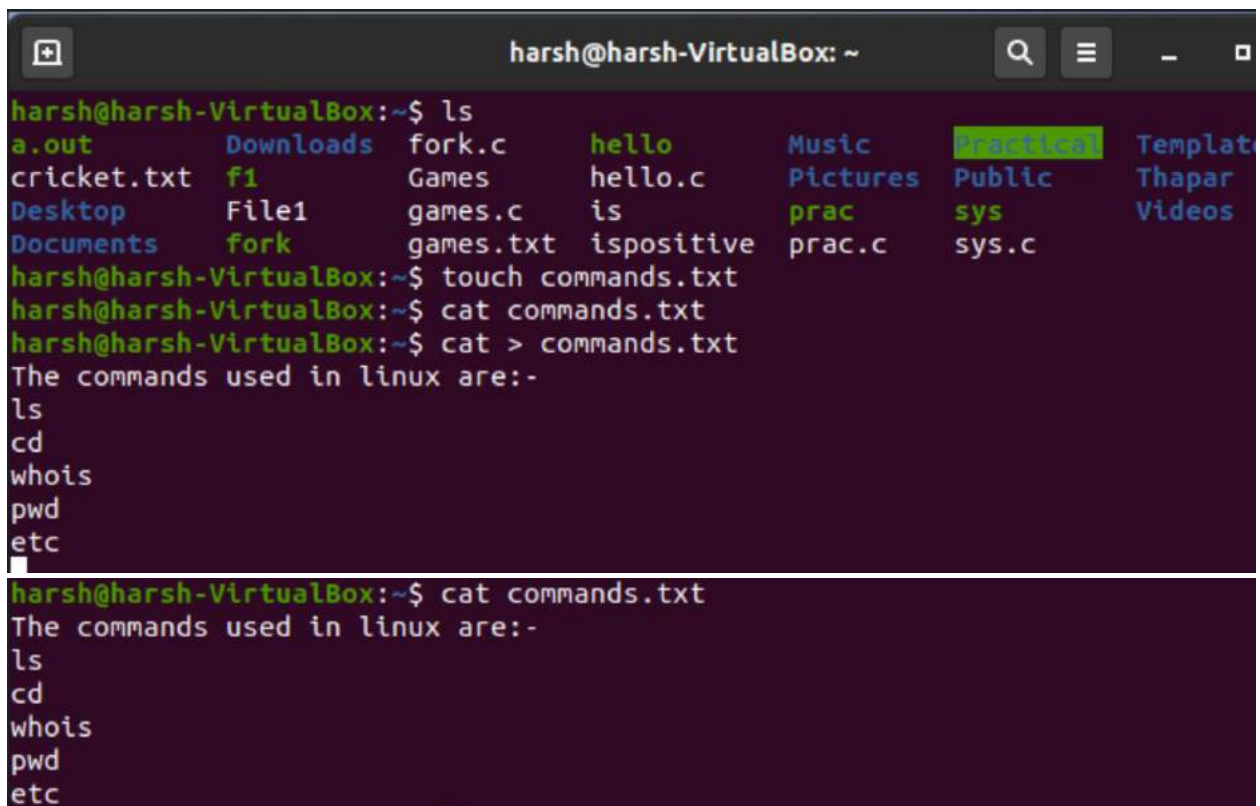
> option for directing the output of the command.

Answer -

Redirection is a feature in Linux such that when executing a command, you can change the standard input/output devices. The basic workflow of any Linux command is that it takes an input and gives an output. The standard input (stdin) device is the keyboard. The standard output (stdout) device is the screen.

The '>' symbol is used for output (STDOUT) redirection.

It is used in cat for adding contents in a file.



```

harsh@harsh-VirtualBox: ~
harsh@harsh-VirtualBox:~$ ls
a.out      Downloads  fork.c     hello      Music      Practical  Template
cricket.txt f1         Games     hello.c    Pictures   Public     Thapar
Desktop    File1     games.c    is         prac       sys        Videos
Documents  fork      games.txt  ispositive prac.c     sys.c
harsh@harsh-VirtualBox:~$ touch commands.txt
harsh@harsh-VirtualBox:~$ cat commands.txt
harsh@harsh-VirtualBox:~$ cat > commands.txt
The commands used in linux are:-
ls
cd
whois
pwd
etc
harsh@harsh-VirtualBox:~$ cat commands.txt
The commands used in linux are:-
ls
cd
whois
pwd
etc

```

Suppose, there are two files, command.txt and linuxcommand.txt. Now, we want to copy the contents of command.tx to linuxcommand.txt, we can use cat command.

```
harsh@harsh-VirtualBox:~$ cat commands.txt
The commands used in linux are:-
ls
cd
whois
pwd
etc
harsh@harsh-VirtualBox:~$ touch linuxcommands.txt
harsh@harsh-VirtualBox:~$ cat commands.txt > linuxcommands.txt
harsh@harsh-VirtualBox:~$ cat linuxcommands.txt
The commands used in linux are:-
ls
cd
whois
pwd
etc
```

It is also used for sorting a file

```
harsh@harsh-VirtualBox:~$ cat commands.txt games.txt > newfile.txt
harsh@harsh-VirtualBox:~$ cat newfile.txt
The commands used in linux are:-
ls
cd
whois
pwd
etc
few more commands are : cd, mkdir, rmdir, rm
Games
hi, let's play a sport
```


This sign is used to overwrite the contents of a file with the output of a command which works as the input of the file.

```
harsh@harsh-VirtualBox:~$ ls
a.out      demo      Downloads  fork      hello     Music     prac.c     sys       Thapar
commands.txt Desktop    f1         fork.c    hello.c   Pictures  Practical  sys.c     ThaparOS
cricket.txt Documents  first      games.txt linuxcommands.txt  prac      Public     Templates Videos
harsh@harsh-VirtualBox:~$ ls > games.txt
harsh@harsh-VirtualBox:~$ cat games.txt
a.out
commands.txt
cricket.txt
demo
Desktop
Documents
Downloads
f1
first
fork
fork.c
games.txt
hello
hello.c
linuxcommands.txt
Music
Pictures
prac
prac.c
Practical
Public
sys
sys.c
Templates
```

Question 5

Introduction and options for the cat command.

Answer -

The cat command is one of the most frequently used command in Linux/Unix, Apple Mac OS X operating systems. cat command allows us to create single or multiple files, view contain the file, concatenate files and redirect output in terminal or files. It is a standard Unix program used to concatenate and display files. The cat command displays file contents to a screen. Cat command concatenate FILE(s), or standard input, to standard output. With no FILE, or when FILE is -, it reads the standard input. Also, you can use cat command for quickly creating a file. The cat command can read and write data from standard input and output devices. It has three

main functions related to manipulating text files: creating them, displaying them, and combining them. The cat command is used for:

- Displaying text file on the screen
- Creating a new text file
- Reading text file
- Modifying file
- File concatenation

Display text file on the screen

```
harsh@harsh-VirtualBox:~$ cat > commands.txt
The commands used in linux are:-
ls
cd
whois
pwd
etc
```

Create a new text file

```
harsh@harsh-VirtualBox:~$ cat > file.txt
hi
hello
```

Appending in file

```
harsh@harsh-VirtualBox:~$ cat commands.txt
The commands used in linux are:-
ls
cd
whois
pwd
etc
harsh@harsh-VirtualBox:~$ cat > commands.txt
few more commands are : cd mkdir, rmdir
```

```

harsh@harsh-VirtualBox:~$ cat >> commands.txt
few more commands are : cd, mkdir, rmdir, rm
harsh@harsh-VirtualBox:~$ cat commands.txt
The commands used in linux are:-
ls
cd
whois
pwd
etc
few more commands are : cd, mkdir, rmdir, rm

```

File concatenation

```

harsh@harsh-VirtualBox:~$ cat commands.txt games.txt >newfile.txt
harsh@harsh-VirtualBox:~$ cat newfile.txt
The commands used in linux are:-
ls
cd
whois
pwd
etc
few more commands are : cd, mkdir, rmdir, rm
Games
hi, let's play a sport

```

Few more are :

Displaying file content with line numbers: cat -n

```

harsh@harsh-VirtualBox:~$ cat -n commands.txt
 1 The commands used in linux are:-
 2 ls
 3 cd
 4 whois
 5 pwd
 6 etc
 7 few more commands are : cd, mkdir, rmdir, rm

```

Displaying '\$' at the end of each line in the file with `cat -e`

```
harsh@harsh-VirtualBox:~$ cat -e commands.txt
The commands used in linux are:-$
ls$
cd$
whois$
pwd$
etc$
few more commands are : cd, mkdir, rmdir, rm$
```

Displaying TAB space in the file with '^I' in the output: `cat -T`

```
harsh@harsh-VirtualBox:~$ cat -T commands.txt
The commands used in linux are:-
ls
cd
whois
pwd
etc
few more commands are : cd, mkdir, rmdir, rm
```

Overwriting a file

```
harsh@harsh-VirtualBox:~$ cat commands.txt games.txt >newfile.txt
harsh@harsh-VirtualBox:~$ cat newfile.txt
The commands used in linux are:-
ls
cd
whois
pwd
etc
few more commands are : cd, mkdir, rmdir, rm
Games
hi, let's play a sport
```

Question 6

Feeding output of command of one command to another by pipeline

Answer -

```
harsh@harsh-VirtualBox:~$ ls -l | more
total 196
-rwxrwxr-x 1 harsh harsh 16776 Aug 18 14:17 a.out
-rw-rw-r-- 1 harsh harsh 98 Aug 25 00:40 commands.txt
-rw-rw-r-- 1 harsh harsh 29 Aug 25 00:49 cricket.txt
drwxr-xr-x 2 harsh harsh 4096 Aug 6 11:33 Desktop
drwxr-xr-x 2 harsh harsh 4096 Aug 2 21:39 Documents
drwxr-xr-x 2 harsh harsh 4096 Aug 23 01:50 Downloads
-rwxrwxrwx 1 harsh root 13 Aug 20 09:11 f1
-rw-rw-r-- 1 harsh harsh 0 Aug 23 04:02 File1
-rw-rw-r-- 1 harsh harsh 9 Aug 25 00:34 file.txt
-rwxrwxr-x 1 harsh harsh 16728 Aug 24 16:40 fork
-rw-rw-r-- 1 harsh harsh 191 Aug 16 12:35 fork.c
-rw-rw-r-- 1 harsh harsh 57 Aug 6 09:11 Games
-rw-rw-r-- 1 harsh harsh 27 Aug 14 10:33 games.c
-rw-rw-r-- 1 harsh harsh 29 Aug 24 15:39 games.txt
-rwxrwxr-x 1 harsh harsh 16728 Aug 14 15:38 hello
-rw-rw-r-- 1 harsh harsh 137 Aug 18 14:40 hello.c
-rw-rw-r-- 1 harsh harsh 0 Aug 20 09:32 is
--More--
```

Pipe is used to combine two or more commands, and in this, the output of one command acts as input to another command, and this command's output may act as input to the next command and so on. This is also used to view the contents of the file in a page by page order. ls commands directly shows the output till the last but when piped with more shows certain directories and file at a time.

Question 7

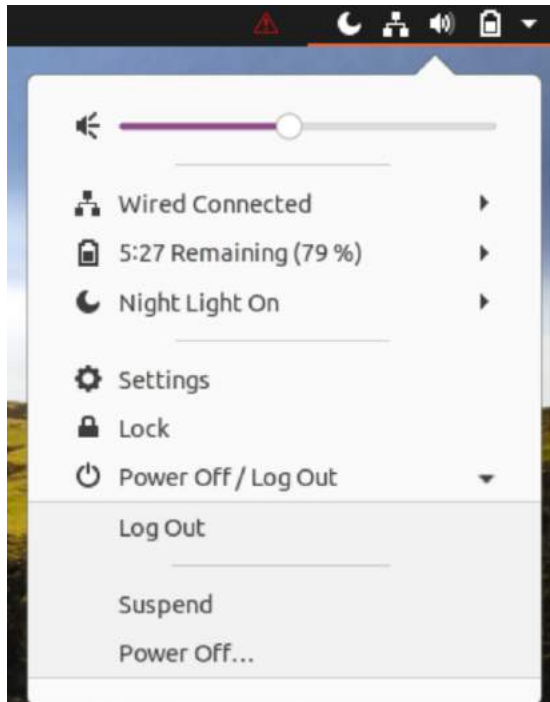
Ways for signing off from linux

Answer -

1. Using the UI
2. Using the Keyboard shortcut
3. Through the application launcher search bar

4. Through the Ubuntu command line, the Terminal

Method 1: Log Out Using The UI

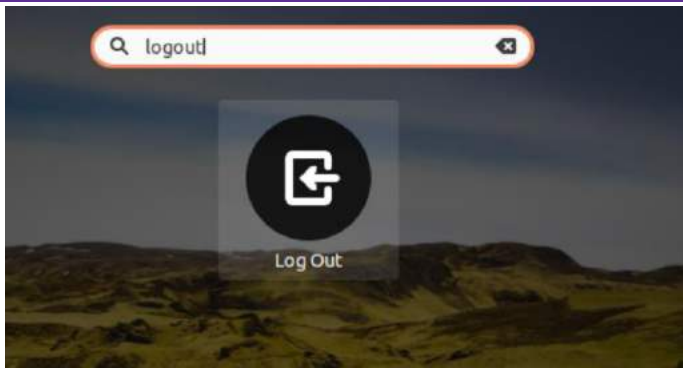


Method 2: Log Out Using Keyboard Shortcut

Ctrl + Alt + Del

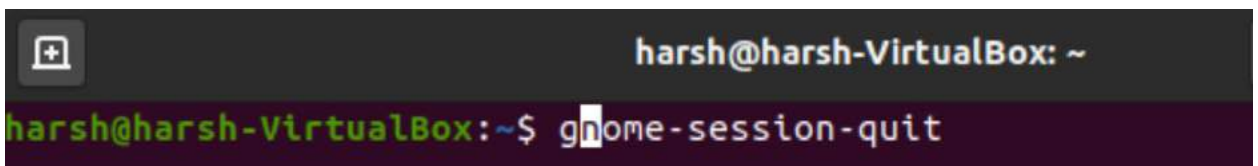


Method 3: Log Out Using the Application Launcher Search



Method 4: Log Out Using the Terminal

gnome-session-quit

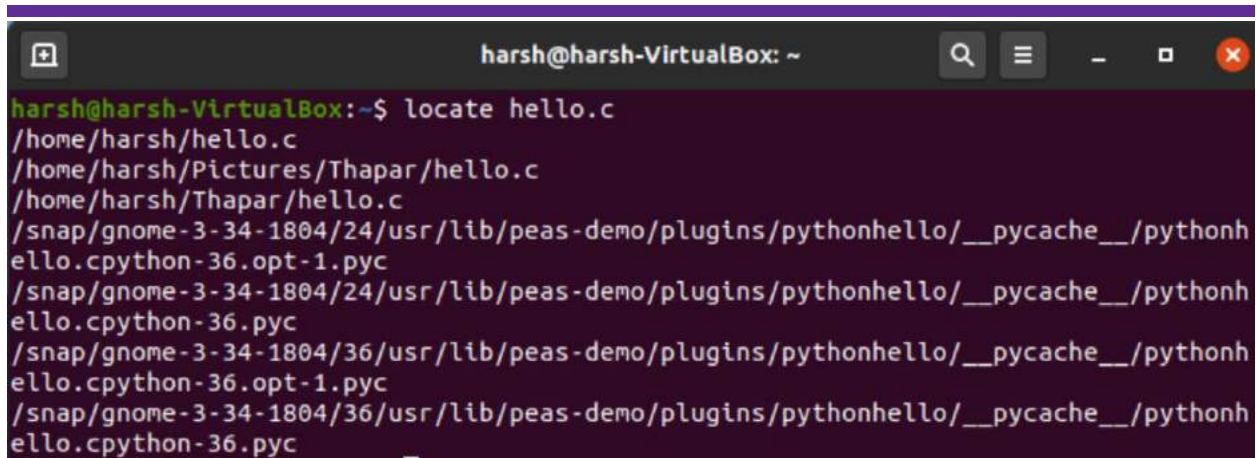


Question 8

Locating commands in Linux.

Answer -

locate command in Linux is used to find the files by name. The locate utility works better and faster as instead of searching the file system when a file search is initiated, it would look through a database.



```

harsh@harsh-VirtualBox: ~
harsh@harsh-VirtualBox:~$ locate hello.c
/home/harsh/hello.c
/home/harsh/Pictures/Thapar/hello.c
/home/harsh/Thapar/hello.c
/snap/gnome-3-34-1804/24/usr/lib/peas-demo/plugins/pythonhello/__pycache__/pythonh
ello.cpython-36.opt-1.pyc
/snap/gnome-3-34-1804/24/usr/lib/peas-demo/plugins/pythonhello/__pycache__/pythonh
ello.cpython-36.pyc
/snap/gnome-3-34-1804/36/usr/lib/peas-demo/plugins/pythonhello/__pycache__/pythonh
ello.cpython-36.opt-1.pyc
/snap/gnome-3-34-1804/36/usr/lib/peas-demo/plugins/pythonhello/__pycache__/pythonh
ello.cpython-36.pyc

```

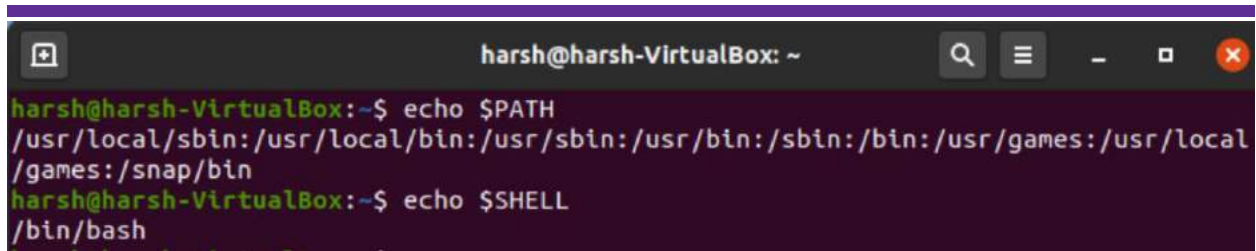
Question 9

PATH and SHELL variable in Linux.

Answer -

The PATH variable is an environment variable that contains an ordered list of paths that Unix will search for executables when running a command. Using these paths means that we do not have to specify an absolute path when running a command. PATH is an environmental variable in Linux and other Unix-like operating systems that tells the shell which directories to search for executable files (i.e., ready-to-run programs) in response to commands issued by a user.

A shell variable is a special variable that is set by the shell and is required by the shell in order to function correctly. Some of these variables are environment variables whereas others are local variables. A shell variable is a variable that is available only to the current shell. ... A shell is the operating system's command interpreter. It processes the commands entered on the command line or read from a shell script file.



```
harsh@harsh-VirtualBox: ~  
harsh@harsh-VirtualBox:~$ echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local  
/games:/snap/bin  
harsh@harsh-VirtualBox:~$ echo $SHELL  
/bin/bash
```

Question 10

Combining the commands in Linux.

Answer -

Linux command chaining is the method of combining several different commands such that each of them can execute in succession based on the operator that separates them. The important part of chaining is the operators that you use to combine them. These operators determine how the commands execute. This set of operators are used mostly to combine different commands and are easy enough to use directly from command line.

1. Logical AND (&&): This command that follows this operator will execute only if the preceding command executes successfully.
2. Logical OR (||): The command that follows will execute only if the preceding command fails.
3. Semi-Colon (;): The succeeding commands will execute regardless of the exit status of the command that precedes it.
4. Pipe (|): The output of the previous command acts as the input to the next command in the chain.
5. Ampersand (&): This sends the current command to the background.
6. Redirection (>, <, >>): The operator can be used to redirect the output of a command or a group of commands to a stream or file.

```
harsh@harsh-VirtualBox: ~/ThaparOS
harsh@harsh-VirtualBox:~$ pwd; ls ; date; who
/home/harsh
a.out      f1      games.c  linuxcommands.txt  prac      Templates
commands.txt  File1   games.txt list.ttx           prac.c    Thapar
cricket.txt  file.txt hello    list.txt           practich  Videos
Desktop      fork    hello.c  Music
Documents    fork.c  is       newfile.txt        Public
Downloads    Games  ispositive Pictures            sys
sys.c
Tuesday 25 August 2020 01:50:58 AM IST
harsh      :0      2020-08-25 01:21 (:0)
harsh@harsh-VirtualBox:~$ mkdir ThaparOS && cd ThaparOS
harsh@harsh-VirtualBox:~/ThaparOS$
harsh@harsh-VirtualBox:~/ThaparOS$ mkdir Play || ls
harsh@harsh-VirtualBox:~/ThaparOS$ ls
Play
harsh@harsh-VirtualBox:~/ThaparOS$
```

```

harsh@harsh-VirtualBox:~$ cat commands.txt | sort commands.txt
cd
etc
few more commands are : cd, mkdir, rmdir, rm
ls
pwd

```

```

harsh@harsh-VirtualBox:~$ ls & date
[1] 4232
Monday 31 August 2020 01:12:06 PM IST
harsh@harsh-VirtualBox:~$ ABC
Desktop      fork.c      linuxcommands.tx
t prac      sys
a.out      Documents  games.txt   multiply    prac.c      sys.c
case      Downloads  hello       Music       practical    Templates
commands.txt f1         hello.c     new         prime       Thapar
cricket.txt first      isnum       new.txt     Public      Videos
demo      fork       ispositive  Pictures    read

```

```

harsh@harsh-VirtualBox:~$ touch new.txt
harsh@harsh-VirtualBox:~$ ls > new.txt
harsh@harsh-VirtualBox:~$ cat new.txt
ABC
a.out
case
commands.txt
cricket.txt
demo
Desktop
Documents
Downloads
f1
first
fork
fork.c
games.txt
hello

```

Question 11

echo and echo -e command

Answer -

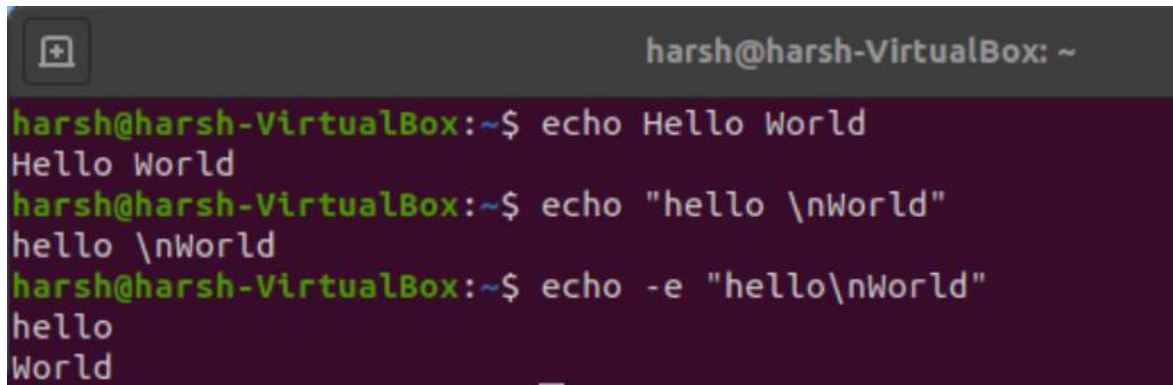
To combine commands in Linux:

- I. Use semicolon operator
- II. Use && operator
- III. Use || operator

echo is a command that outputs the strings it is being passed as arguments. It is a command available in various operating system shells and typically used in shell scripts and batch files to output status text to the screen or a computer file, or as a source part of a pipeline. Echo is a Unix/Linux command tool used for displaying lines of text or string which are passed as arguments on the command line. This command is most commonly used in shell scripts.

The -e parameter is used for the interpretation of backslash interpreters. echo -e helps the terminal of Linux understand the escape sequences.

Operation of both echo and echo -e is listed below.

A terminal window titled 'harsh@harsh-VirtualBox: ~' showing three commands and their outputs. The first command is 'echo Hello World' which outputs 'Hello World'. The second command is 'echo "hello \nWorld"' which outputs 'hello \nWorld'. The third command is 'echo -e "hello\nWorld"' which outputs 'hello' followed by a new line and then 'World'.

```
harsh@harsh-VirtualBox:~$ echo Hello World
Hello World
harsh@harsh-VirtualBox:~$ echo "hello \nWorld"
hello \nWorld
harsh@harsh-VirtualBox:~$ echo -e "hello\nWorld"
hello
World
```

Question 12

cal and date command with its different operations.

Answer -

cal command is a calendar command in Linux which is used to see the calendar of a specific month or a whole year.

- **cal** : Shows current month calendar on the terminal.
- **cal month year**: Shows calendar of selected month and year.

```
harsh@harsh-VirtualBox:~$ cal
      August 2020
Su Mo Tu We Th Fr Sa
                1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
harsh@harsh-VirtualBox:~$ cal 3 2020
      March 2020
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

- **cal year** : Shows the whole calendar of the year.

```

harsh@harsh-VirtualBox:~$ cal 2020
                2020
    January      February      March
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
   1  2  3  4           1  1  2  3  4  5  6  7
  5  6  7  8  9 10 11  2  3  4  5  6  7  8  8  9 10 11 12 13 14
12 13 14 15 16 17 18  9 10 11 12 13 14 15 15 16 17 18 19 20 21
19 20 21 22 23 24 25 16 17 18 19 20 21 22 22 23 24 25 26 27 28
26 27 28 29 30 31    23 24 25 26 27 28 29 29 30 31

    April        May          June
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
   1  2  3  4           1  2           1  2  3  4  5  6
  5  6  7  8  9 10 11  3  4  5  6  7  8  9  7  8  9 10 11 12 13
12 13 14 15 16 17 18 10 11 12 13 14 15 16 14 15 16 17 18 19 20
19 20 21 22 23 24 25 17 18 19 20 21 22 23 21 22 23 24 25 26 27
26 27 28 29 30      24 25 26 27 28 29 30 28 29 30
                   31

    July          August       September
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
   1  2  3  4           1           1  2  3  4  5
  5  6  7  8  9 10 11  2  3  4  5  6  7  8  6  7  8  9 10 11 12
12 13 14 15 16 17 18  9 10 11 12 13 14 15 13 14 15 16 17 18 19
19 20 21 22 23 24 25 16 17 18 19 20 21 22 20 21 22 23 24 25 26
26 27 28 29 30 31    23 24 25 26 27 28 29 27 28 29 30
                   30 31

    October       November     December
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
   1  2  3           1  2  3  4  5  6  7           1  2  3  4  5
  4  5  6  7  8  9 10  8  9 10 11 12 13 14  6  7  8  9 10 11 12
11 12 13 14 15 16 17 15 16 17 18 19 20 21 13 14 15 16 17 18 19
18 19 20 21 22 23 24 22 23 24 25 26 27 28 20 21 22 23 24 25 26
25 26 27 28 29 30 31 29 30      27 28 29 30 31

```

- cal -3 : Shows calendar of previous, current and next month

```

harsh@harsh-VirtualBox:~$ cal -3
                2020
    July          August       September
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
   1  2  3  4           1           1  2  3  4  5
  5  6  7  8  9 10 11  2  3  4  5  6  7  8  6  7  8  9 10 11 12
12 13 14 15 16 17 18  9 10 11 12 13 14 15 13 14 15 16 17 18 19
19 20 21 22 23 24 25 16 17 18 19 20 21 22 20 21 22 23 24 25 26
26 27 28 29 30 31    23 24 25 26 27 28 29 27 28 29 30
                   30 31

```

date command is used to display the system date and time. date command is also used to set the date and time of the system. By default, the date command displays the date in the time zone on which Unix/Linux operating system is configured. You must be the super-user (root) to change the date and time.

- date: the date command displays the current date and time, of the current time zone.

- `date -u`: Displays the time in GMT(Greenwich Mean Time)/UTC(Coordinated Universal Time)time zone.
- `-date` or `-d` Option:
 - a. Displays the given date string in the format of date.
 - b. Displays date of past.
 - c. Displays date of future.
- `date -s = " " :` To set a new date.

```
harsh@harsh-VirtualBox:~$ date
Tuesday 25 August 2020 02:28:33 AM IST
harsh@harsh-VirtualBox:~$ date -u
Monday 24 August 2020 08:59:08 PM UTC
harsh@harsh-VirtualBox:~$ date --date="Aug 24 2020"
Monday 24 August 2020 12:00:00 AM IST
harsh@harsh-VirtualBox:~$ date --date="10 years ago"
Wednesday 25 August 2010 02:31:57 AM IST
harsh@harsh-VirtualBox:~$ date --date="10 years"
Sunday 25 August 2030 02:32:20 AM IST
```

Question 13

man and help command

Answer -

man command in Linux is used to display the user manual of any command that we can run on the terminal. It provides a detailed view of the command. The man command gives users access to manual pages for command-line utilities and tools. man command in Linux is used to display the user manual of any command that we can run on the terminal. It provides a detailed view of the command which includes Name, Synopsis, Description, Options, Exit Status, Return Values, Errors, Files, Versions, Examples, Authors and See also

```
harsh@harsh-VirtualBox:~$ man mkdir
```

```

MKDIR(1)                                User Commands                                MKDIR(1)

NAME
    mkdir - make directories

SYNOPSIS
    mkdir [OPTION]... DIRECTORY...

DESCRIPTION
    Create the DIRECTORY(ies), if they do not already exist.

    Mandatory arguments to long options are mandatory for short options too.

    -m, --mode=MODE
        set file mode (as in chmod), not a=rwx - umask

    -p, --parents
        no error if existing, make parent directories as needed

    -v, --verbose
        print a message for each created directory

    -Z      set SELinux security context of each created directory to the de-
            fault type

    --context[=CTX]

Manual page mkdir(1) line 1 (press h for help or q to quit)

```

help command helps you to learn about any built-in command. help command just displays information about shell built-in commands. If we don't know how to use a command i.e. we don't know about its parameters and return type etc, then we can make use of `-h` or `-help` command.


```

harsh@harsh-VirtualBox:~$ mkdir --help
Usage: mkdir [OPTION]... DIRECTORY...
Create the DIRECTORY(ies), if they do not already exist.

Mandatory arguments to long options are mandatory for short options too.
-m, --mode=MODE    set file mode (as in chmod), not a=rwx - umask
-p, --parents       no error if existing, make parent directories as needed
-v, --verbose       print a message for each created directory
-Z                set SELinux security context of each created directory
                  to the default type
--context[=CTX]    like -Z, or if CTX is specified then set the SELinux
                  or SMACK security context to CTX
--help            display this help and exit
--version         output version information and exit

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Full documentation at: <https://www.gnu.org/software/coreutils/mkdir>
or available locally via: info '(coreutils) mkdir invocation'

```

Question 14

Using escape sequences

Answer -

Escape sequences are non-printable characters that are used for formatted output. Examples are `\n` for a new line, `\t` for horizontal tab, `\` for double quotes etc. An escape sequence is two or more characters that often begin with an escape character that tells the computer or software program to perform a function or command. Examples- `\`, `\'`, `\a`, `\b`, `\n`, `\f`, etc.

```

harsh@harsh-VirtualBox:~$ echo -e "Hello\nWorld\tThapar\"Good Morning\" "
Hello
World  Thapar"Good Morning"

```

Question 15

print command in Linux

Answer -

“printf” command in Linux is used to display the given string, number or any other format specifier on the terminal window. It works the same way as “printf” works in programming languages like C.

```
harsh@harsh-VirtualBox:~$ printf "%s" "Hello Thapar"
Hello Thaparharsh@harsh-VirtualBox:~$ printf "%b\n" "Hello Thapar"
Hello Thapar
```

Assignment 3

Question 1

Internal and External Commands.

Answer -

Internal Commands:

- Commands which are built into the shell.
- For all the shell built-in commands, execution of the same is fast in the sense that the shell doesn't have to search the given path for them in the PATH variable, and also no process needs to be spawned for executing it.
- Internal commands are commands that are already loaded in the system.
- They can be executed at any time and are independent.
- Internal commands are a part of the shell. No separate process is necessary to run these.
- Examples: source, cd, fg, etc.

External Commands:

- Commands which aren't built into the shell.
- When an external command has to be executed, the shell looks for its path given in the PATH variable, and also a new process has to be spawned and the command gets executed.
- External commands are loaded when the user requests for them.
- External commands will have an individual process.

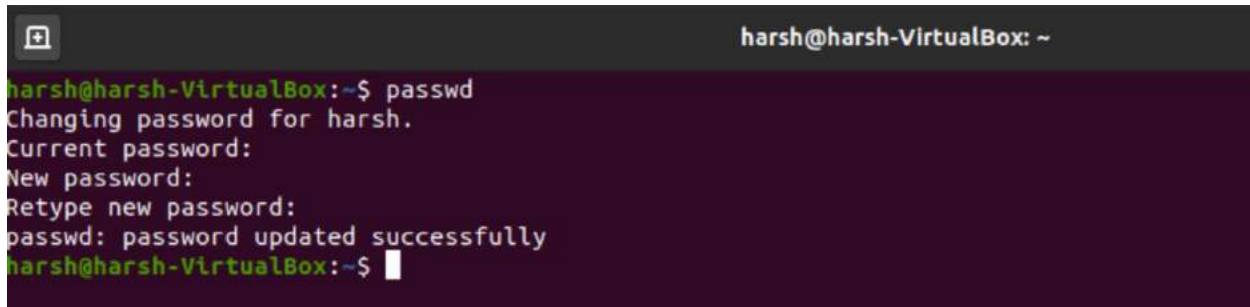
- They are usually located in /bin or /usr/bin. For example, when you execute the “cat” command, which usually is at /usr/bin, the executable /usr/bin/cat gets executed.
 - The files for the command not present in the path, the external command won't be executed. In order to run these commands, a separate process id is required.
 - Examples: ls, cat etc.
-

Question 2

Commands - passwd, who, uname, tty, stty

Solution

passwd : On Unix-like operating systems, the passwd command is used to change the password of a user account. A normal user can run passwd to change their own password, and a system administrator (the superuser) can use passwd to change another user's password, or define how that account's password can be used or changed.

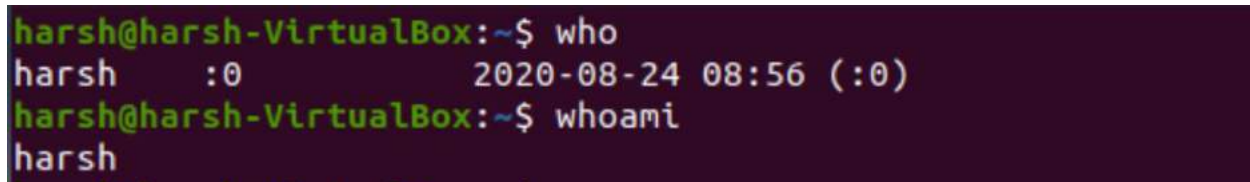


```

harsh@harsh-VirtualBox: ~
harsh@harsh-VirtualBox:~$ passwd
Changing password for harsh.
Current password:
New password:
Retype new password:
passwd: password updated successfully
harsh@harsh-VirtualBox:~$

```

who : who command is used to find out the time of last system boot, current run level of the system, list of logged-in users and more. The who command is used to get information about currently logged in user on to the system.

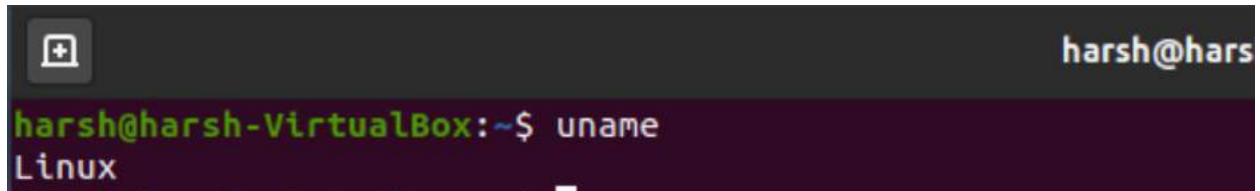


```

harsh@harsh-VirtualBox:~$ who
harsh      :0                2020-08-24 08:56 (:0)
harsh@harsh-VirtualBox:~$ whoami
harsh

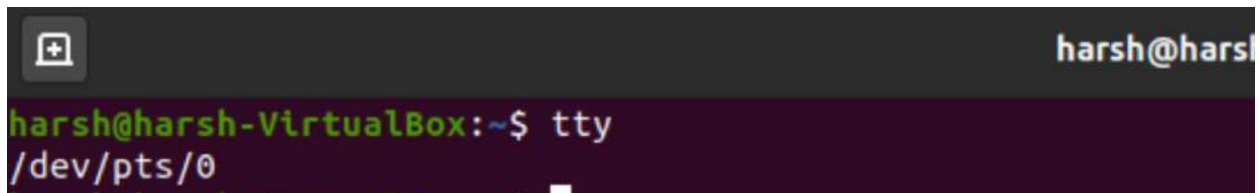
```

uname: The command '*uname*' displays the information about the system. The *uname* tool is most commonly used to determine the processor architecture, the system hostname and the version of the kernel running on the system.

A terminal window with a dark background. The prompt is 'harsh@harsh-VirtualBox:~\$'. The command 'uname' has been entered, and the output 'Linux' is displayed on the next line.

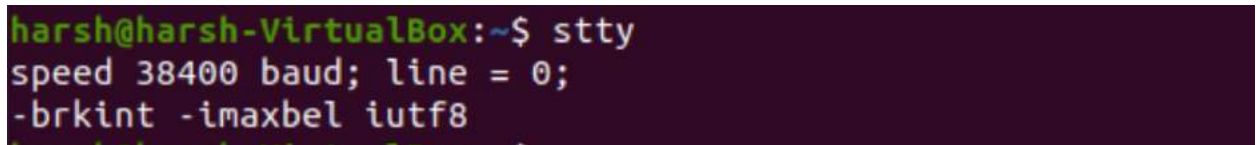
```
harsh@harsh-VirtualBox:~$ uname
Linux
```

tty : The *tty* command will print the name of the device file that your pseudo-teletype slave is using to interface to the master. And that, effectively, is the number of your terminal window.

A terminal window with a dark background. The prompt is 'harsh@harsh-VirtualBox:~\$'. The command 'tty' has been entered, and the output '/dev/pts/0' is displayed on the next line.

```
harsh@harsh-VirtualBox:~$ tty
/dev/pts/0
```

stty : *stty* command in Linux is used to change and print terminal line settings. Basically, this command shows or changes terminal characteristics.

A terminal window with a dark background. The prompt is 'harsh@harsh-VirtualBox:~\$'. The command 'stty' has been entered, and the output 'speed 38400 baud; line = 0; -brkint -imaxbel iutf8' is displayed on the next line.

```
harsh@harsh-VirtualBox:~$ stty
speed 38400 baud; line = 0;
-brkint -imaxbel iutf8
```

Question 3

Types of file and file system in Linux.

Answer

ls command will show the file type as an encoded symbol found as the first character of the file permission part. In this case, it is "-", which means "regular file". It is important to point out that Linux file types are not to be mistaken with file extensions. Let us have a look at a short summary of all the seven different types of Linux file types and ls command identifiers:

1. - : regular file:
 - most common file type found on the Linux system.
 - governs all different files such as text files, images, binary files, shared libraries, etc.
 - Can be created as a regular file with the touch command.
 - rm for removing the file.
2. d : directory
 - second most common file type found in Linux.
 - can be created with the mkdir command.
 - Rmdir for removing the directory.
3. c : character device file
 - allow users and programs to communicate with hardware peripheral devices.
4. b : block device file
 - similar to character devices.
 - mostly govern hardware as hard drives, memory, etc.
5. s : local socket file
 - are used for communication between processes.
 - used by services such as X windows, syslog and etc.
6. p : named pipe
 - allow communication between two local processes.
 - can be created by the mknod command and removed with the rm command.
7. l : symbolic link
 - an administrator can assign a file or directory multiple identities.
 - can be though a pointer to an original file.

Linux File System or any file system generally is a layer which is under the operating system that handles the positioning of your data on the storage, without it; the system cannot know which file starts from where and ends where it ends.

Linux offers many file systems like these: Ext, Ext2, Ext3, Ext4, JFS, XFS, btrfs and swap

1. Ext: an old one and no longer used due to limitations.
 2. Ext2: first Linux file system that allows 2 terabytes of data allowed.
 3. Ext3: It came from Ext2, but with upgrades and backward compatibility. The only problem about it is that the servers don't use this kind of file system because this file system doesn't support file recovery or disk snapshots.
 4. Ext4: faster and allow large files with significant speed. It is a very good option for SSD disks and you notice when you try to install any Linux distro that this one is the default file system that Linux suggests.
 5. JFS: old file system made by IBM. It works very well with small and big files, but it failed and files corrupted after long time use, reports say.
 6. XFS: old file system and works slowly with small files.
 7. Btrfs: made by Oracle. It is not as stable as Ext in some distros, but you can say that it is a replacement for it if you have to. It has a good performance
-

Question 4

Directory structure in Linux

Solution -

The Filesystem Hierarchy Standard (FHS) defines the structure of file systems on Linux and other UNIX-like operating systems.

1. / – The Root Directory
 - Every single file and directory starts from the root directory.
 - Only the root user has write privilege under this directory.
2. /bin – Essential User Binaries
 - Contains binary executables.
 - Common linux commands you need to use in single-user modes are located under this directory.
3. /sbin – System Binaries
 - Just like /bin, /sbin also contains binary executables typically by system administrators.

-
4. /etc – Configuration Files
 - Contains configuration files required by all programs.
 - This also contains startup and shutdown shell scripts used to start/stop individual programs.
 5. /dev – Device Files
 - Contains device files.
 6. /proc – Process Information
 - Contains information about system process.
 7. /var – Variable Files
 - var stands for variable files.
 8. /tmp – Temporary Files
 9. /usr – User Programs
 - Contains binaries, libraries, documentation, and source-code for second level programs.
 10. /home – Home Directories
 - Home directories for all users to store their personal files.
 11. /boot – Boot Loader Files
 - Contains boot loader related files.
 12. /lib – System Libraries
 - Contains library files that supports the binaries located under /bin and /sbin
 13. /opt – Optional add-on Applications
 - opt stands for optional.
 - Contains add-on applications from individual vendors.
 14. /mnt – Mount Directory
 - Temporary mount directory where sysadmins can mount filesystems.
 15. /media – Removable Media Devices
 - Temporary mount directory for removable devices.
 16. /srv – Service Data
 - srv stands for service.

Using tree command to show structure of directory.

```
harsh@harsh-VirtualBox:~$ cd /
harsh@harsh-VirtualBox:/$ tree -L 1
.
├── bin -> usr/bin
├── boot
├── cdrom
├── dev
├── etc
├── home
├── lib -> usr/lib
├── lib32 -> usr/lib32
├── lib64 -> usr/lib64
├── libx32 -> usr/libx32
├── lost+found
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin -> usr/sbin
├── snap
├── srv
├── swapfile
├── sys
├── tmp
├── usr
└── var
```

Question 5

HOME variable in Linux

Solution -

HOME contains the path to the **home** directory of the current user. This **variable** can be used by applications to associate configuration files and such like with the user running it.

```
harsh@harsh-VirtualBox:~$ echo $HOME
/home/harsh
harsh@harsh-VirtualBox:~$
```

Question 6

Commands - pwd, cd, mkdir, rmdir,

Solution -pwd stands for Print Working Directory. It prints the path of the working directory, starting from the root. pwd is shell built-in command(pwd) or an actual binary(/bin/pwd).

cd (“change directory”) command is used to change the current working directory in Linux and other Unix-like operating systems.

mkdir command is used to create new directories. A directory, referred to as a folder in some operating systems, appears to the user as a container for other directories and files.

rmdir command is used to remove empty directories from the filesystem in Linux. The rmdir command removes each and every directory specified in the command line only if these directories are empty.

```

harsh@harsh-VirtualBox:~$ pwd
/home/harsh
harsh@harsh-VirtualBox:~$ ls
a.out      f1      Games  is      prac      sys      Videos
Desktop    File1   games.c ispositive prac.c    sys.c
Documents  fork    hello  Music   Practical Templates
Downloads  fork.c  hello.c Pictures Public    Thapar
harsh@harsh-VirtualBox:~$ cd Downloads
harsh@harsh-VirtualBox:~/Downloads$ ls
alex-mustaros-KiXSWo3cEAI-unsplash.jpg  denys-nevozhai-7f0U7gCulR4-unsplash.jpg
brooke-lark-YSA1IRkGAsg-unsplash.jpg
harsh@harsh-VirtualBox:~/Downloads$ cd ../
harsh@harsh-VirtualBox:/$ cd ~
harsh@harsh-VirtualBox:~$ mkdir OS
harsh@harsh-VirtualBox:~$ ls
a.out      f1      Games  is      Pictures  Public  Thapar
Desktop    File1   games.c ispositive prac      sys      Videos
Documents  fork    hello  Music   prac.c    sys.c
Downloads  fork.c  hello.c OS        Practical Templates
harsh@harsh-VirtualBox:~$ rmdir OS
harsh@harsh-VirtualBox:~$ ls
a.out      f1      Games  is      prac      sys      Videos
Desktop    File1   games.c ispositive prac.c    sys.c
Documents  fork    hello  Music   Practical Templates
Downloads  fork.c  hello.c Pictures Public    Thapar
harsh@harsh-VirtualBox:~$

```

Question 7

Absolute and relative pathname

Solution -

Absolute Path

An absolute path is defined as the specifying the location of a file or directory from the root directory(/).

Starts at the root directory (/) and works down and a slash (/) after every directory name

```
harsh@harsh-VirtualBox:~$ pwd
/home/harsh
harsh@harsh-VirtualBox:~$ ls
a.out      f1      Games   is      prac     sys      Videos
Desktop    File1   games.c ispositive  prac.c   sys.c
Documents  fork    hello   Music   Practical Templates
Downloads  fork.c  hello.c Pictures Public    Thapar
harsh@harsh-VirtualBox:~$ cd /home/harsh/Practical
harsh@harsh-VirtualBox:~/Practical$ cd /home/harsh
harsh@harsh-VirtualBox:~$
```

Relative path

Relative path is defined as the path related to the present working directory(pwd). It starts at your current directory and never starts with a /.

```

harsh@harsh-VirtualBox:~$ ls
a.out      f1      Games   is      prac     sys      Videos
Desktop    File1   games.c ispositive prac.c   sys.c
Documents  fork    hello   Music   Practical Templates
Downloads  fork.c  hello.c Pictures Public    Thapar
harsh@harsh-VirtualBox:~$ cd Pictures
harsh@harsh-VirtualBox:~/Pictures$ cd ..
harsh@harsh-VirtualBox:~$

```

Question 8

Using . and ..

Solution -

The two dots here are used in cd command. It is used for going back one step from a directory.

```

harsh@harsh-VirtualBox:~$ cd Desktop
harsh@harsh-VirtualBox:~/Desktop$ cd ..

```

Single dots are used for find operation- to find a file with name test inside the current directory and all directories inside the current directory.

```

harsh@harsh-VirtualBox:~$ ls
a.out      f1      Games   is      prac     sys      Videos
Desktop    File1   games.c ispositive prac.c   sys.c
Documents  fork    hello   Music   Practical Templates
Downloads  fork.c  hello.c Pictures Public    Thapar
harsh@harsh-VirtualBox:~$ find . -name "F"
harsh@harsh-VirtualBox:~$ find . -name "fork"
./fork

```

Question 9

Commands - Cat, cp, rm, mv, wc, comm, cmp, diff

Solution -

cat - The cat is one of the most frequently used commands in Linux like operating systems. cat command allows us to create single or multiple files, view content of file, concatenate files and redirect output in terminal or files. cat is a standard Unix utility that reads files sequentially, writing them to standard output. The name is derived from its function to concatenate files.

```
harsh@harsh-VirtualBox:~$ ls
a.out      f1      Games   is      prac     sys      Videos
Desktop    File1   games.c ispositive prac.c   sys.c
Documents  fork    hello   Music   Practical Templates
Downloads  fork.c  hello.c Pictures Public    Thapar

harsh@harsh-VirtualBox:~$ touch games.txt
harsh@harsh-VirtualBox:~$ cat >> games.txt
Hi, let's play a sport.
cricket.

harsh@harsh-VirtualBox:~$ cat games.txt
Hi, let's play a sport.
cricket.
```

cp - The cp command is a command-line utility for copying files and directories. It supports moving one or more files or folders with options for taking backups and preserving attributes. Copies of files are independent of the original file unlike the mv command. This command is used to copy files or groups of files or directory. It creates an exact image of a file on a disk with a different file name.

```
harsh@harsh-VirtualBox:~$ cat games.txt
Hi, let's play a sport.
cricket.
harsh@harsh-VirtualBox:~$ touch cricket.txt
harsh@harsh-VirtualBox:~$ cp games.txt cricket.txt
harsh@harsh-VirtualBox:~$ cat cricket.txt
Hi, let's play a sport.
cricket.
harsh@harsh-VirtualBox:~$
```

rm - rm command is used to remove objects such as files, directories, symbolic links and so on from the file system like UNIX.

```
harsh@harsh-VirtualBox:~$ ls
a.out      Downloads  fork.c     hello      Music      Practical  Templates
cricket.txt f1         Games     hello.c    Pictures   Public     Thapar
Desktop    File1      games.c    is         prac       sys        Videos
Documents  fork      games.txt ispositive prac.c     sys.c
harsh@harsh-VirtualBox:~$ rm games.txt
harsh@harsh-VirtualBox:~$ ls
a.out      Downloads  fork.c     hello.c    Pictures   Public     Thapar
cricket.txt f1         Games     is         prac       sys        Videos
Desktop    File1      games.c    ispositive prac.c     sys.c
Documents  fork      hello      Music      Practical  Templates
harsh@harsh-VirtualBox:~$
```

mv -The mv command is a command line utility that moves files or directories from one place to another . It supports moving single files, multiple files and directories. It can prompt before

overwriting and has an option to only move files that are new than the destination.

```
harsh@harsh-VirtualBox:~$ ls
a.out      Downloads  fork.c    hello.c   Pictures  Public    Thapar
cricket.txt f1         Games    is        prac      sys       Videos
Desktop    File1     games.c  ispositive prac.c    sys.c
Documents  fork      hello    Music     Practico Templates
harsh@harsh-VirtualBox:~$ mv cricket.txt Thapar
harsh@harsh-VirtualBox:~$ ls Thapar
Bhai.txt  cricket.txt  Games1  Games12  Games122  hello.c
harsh@harsh-VirtualBox:~$
```

wc -The wc (word count) command in Unix/Linux operating systems is used to find out number of newline count, word count, byte and characters count in a files specified by the file arguments.

```
harsh@harsh-VirtualBox:~/Thapar$ wc cricket.txt
 2  6 33 cricket.txt
```

comm- The comm command in the Unix family of computer operating systems is a utility that is used to compare two files for common and distinct lines.

```
harsh@harsh-VirtualBox:~$ cat games.txt
Games
hi, let's play a sport
harsh@harsh-VirtualBox:~$ cat cricket.txt
Hi, let's play a sport.
cricket.
harsh@harsh-VirtualBox:~$ comm games.txt cricket.txt
Games
hi, let's play a sport
      Hi, let's play a sport.
comm: file 2 is not in sorted order
      cricket.
harsh@harsh-VirtualBox:~$
```

cmp - cmp is a command-line utility for computer systems that use Unix or a Unix-like operating system. It compares two files of any type and writes the results to the standard output.

diff - diff is a command-line utility that allows you to compare two files line by line. It can also compare the contents of directories. The diff command is most commonly used to create a patch containing the differences between one or more files that can be applied using the patch command.

```
harsh@harsh-VirtualBox:~$ ls
a.out      Downloads  fork.c     hello      Music      Practical  Templates
cricket.txt f1         Games     hello.c    Pictures   Public     Thapar
Desktop    File1      games.c    is         prac       sys        Videos
Documents  fork       games.txt ispositive prac.c     sys.c
harsh@harsh-VirtualBox:~$ diff Desktop Thapar
Only in Thapar: Bhai.txt
Only in Thapar: Games1
Only in Thapar: Games12
Only in Thapar: Games122
Only in Thapar: hello.c
harsh@harsh-VirtualBox:~$
```

Question 9

Compressing and archiving files (zip, tar)

Solution -

An archive file is a collection of files and directories stored in one file. The archive file is not compressed — it uses the same amount of disk space as all the individual files and directories combined.

On the other hand, a compressed file is a collection of files and directories that are stored in one file and stored in a way that uses less disk space than all the individual files and directories combined. 'zip' is a command-line utility that helps you create Zip archives.

The zip command takes the following syntax form:- zip OPTIONS ARCHIVE_NAME FILES Zip files do not support Linux-style ownership information. The extracted files are owned by the user that runs the command.

To preserve the file ownership and permissions use the "tar" command. The "tar" is most widely used command to create compressed archive files and that can be moved easily from one disk to another disk or machine to machine.

The Linux "tar" stands for tape archive, which is used by large number of Linux/Unix system administrators to deal with tape drives backup. The tar command used to rip a collection of files and directories into highly compressed archive file commonly called tarball or tar, gzip and bzip in Linux. The tar is most widely used command to create compressed archive files and that can be moved easily from one disk to another disk or machine to machine.

```
harsh@harsh-VirtualBox:~$ tar -cvf Thapar.tar Desktop
Desktop/
harsh@harsh-VirtualBox:~$ ls
a.out      Desktop  fork.c    linuxcommands.txt  prac      sys
case       Documents games.txt  multiply            prac.c    sys.c
commands.txt Downloads hello      Music              practical  Templates
cricket.txt f1       hello.c    new                prime     Thapar
demo       first   isnum      new.txt           Public    Thapar.tar
fork       ispositive Pictures    read            Videos
```

ZIP is a compression and file packaging utility for Unix. Each file is stored in a single .zip {zip-filename} file with the extension .zip. zip is used to compress the files to reduce file size and also used as file package utility. zip is available in many operating systems like unix, linux, windows etc.

HARSH KASHYAP

CSE 4

101917088

hkashyap_be19@thapar.edu

A Practical activity Report submitted
for Operating Systems (UCS303)

OPERATING SYSTEM



Computer Science and Engineering
Patiala Campus
2020

Submitted to
Dr. Ashwani Kumar

Assignment 4

Question 1

Basic file attributes : ls -l, -d option.

Answer -

Using ls -l character will display a long listing of the content of current directory (i.e. not only prints the name of the file, but also some attributes such as owner, group owner, link count, permissions).

The first character shows the file type. The next nine characters are showing the file permissions. The first three characters are for the user, the next three are for the group, and the last three are for others.

```

harsh@harsh-VirtualBox:~$ ls -l
total 208
-rwxrwxr-x 1 harsh harsh 16776 Aug 18 14:17 a.out
-rwxrwxr-x 1 harsh harsh 95 Aug 31 00:16 case
-rw-rw-r-- 1 harsh harsh 98 Aug 25 00:40 commands.txt
-rw-rw-r-- 1 harsh harsh 29 Aug 25 00:49 cricket.txt
-rwxrwxrwx 1 harsh harsh 84 Aug 30 17:13 demo
drwxr-xr-x 2 harsh harsh 4096 Aug 6 11:33 Desktop
drwxr-xr-x 2 harsh harsh 4096 Aug 2 21:39 Documents
drwxr-xr-x 2 harsh harsh 4096 Aug 23 01:50 Downloads
-rwxrwxrwx 1 harsh root 13 Aug 20 09:11 f1
-rwxrwxr-x 1 harsh harsh 33 Aug 28 13:11 first
-rwxrwxr-x 1 harsh harsh 16728 Aug 24 16:40 fork
-rw-rw-r-- 1 harsh harsh 191 Aug 16 12:35 fork.c
-rw-rw-r-- 1 harsh harsh 188 Aug 31 01:52 games.txt
-rwxrwxr-x 1 harsh harsh 16728 Aug 14 15:38 hello
-rw-rw-r-- 1 harsh harsh 137 Aug 18 14:40 hello.c
-rwxrwxr-x 1 harsh harsh 44 Aug 30 22:52 isnum
-rwxrwxr-x 1 harsh harsh 45 Aug 30 22:55 ispositive
-rw-rw-r-- 1 harsh harsh 53 Aug 25 00:18 linuxcommands.txt
-rwxrwxr-x 1 harsh harsh 236 Aug 30 23:08 multiply
drwxr-xr-x 2 harsh harsh 4096 Aug 2 21:39 Music
-rwxrwxr-x 1 harsh harsh 173 Aug 30 16:19 new

```

Using ls -d character will display a long listing of the content of the directories present without showing the contents of the directory. This command is used to list only directories in the current directory. We have to type "ls -d */" command in the command window

```

harsh@harsh-VirtualBox:~$ ls -d */
Desktop/  Downloads/  Pictures/  Public/  Videos/
Documents/ Music/      Practises/  Templates/
harsh@harsh-VirtualBox:~$

```

Question 2

File Permission and Changing the access right.

Answer -

In Linux operating systems, chmod is the command and system call which is used to change the access permissions of file system objects. It is used in other machines like servers, mainframe computers, supercomputers, etc.

Every file and directory in your Linux system has following 3 permissions defined for all the owners-

1. Read: This permission gives us the authority to open and read a file. Read permission on a directory gives you the ability to list its content.
2. Write: The write permission gives us the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory.
3. Execute: In Linux, we cannot run a program unless the execute permission is set.

Changing access rights - Using the "chmod" command, we can set permissions (read, write, execute) on a file/directory for the owner, group and the world. Syntax - chmod [reference][operator][mode] file.

The user can

- read, write, and execute it;
- members of your group can read and execute it,
- others may only read it.

chmod u=rwx,g=rx,o=r myfile, letters u, g, and o stand for "user", "group", and "other".

We can also assign commands to files using numbers, each digit being a combination of the numbers 4, 2, 1, and 0:

- 4 stands for "read",
- 2 stands for "write",
- 1 stands for "execute",
- 0 stands for "no permission."

Removing all permission from cricket.txt which has read, write permission for user, read and write for a group and read for others.


```

harsh@harsh-VirtualBox:~$ ls -l cricket.txt
-rw-rw-r-- 1 harsh harsh 29 Aug 25 00:49 cricket.txt
harsh@harsh-VirtualBox:~$ chmod ugo-wxr cricket.txt
harsh@harsh-VirtualBox:~$ ls -l cricket.txt
----- 1 harsh harsh 29 Aug 25 00:49 cricket.txt
harsh@harsh-VirtualBox:~$ 

```

Question 3

Relative and absolute permission

Answer -

Absolute mode - In this mode, file permissions are not represented as characters but a three-digit octal number. So we use numbers to represent file permissions. This is the method which is most commonly used to set permissions.

```

harsh@harsh-VirtualBox:~$ ls -l cricket.txt
----- 1 harsh harsh 29 Aug 25 00:49 cricket.txt
harsh@harsh-VirtualBox:~$ chmod 777 cricket.txt
harsh@harsh-VirtualBox:~$ ls -l cricket.txt
-rwxrwxrwx 1 harsh harsh 29 Aug 25 00:49 cricket.txt
harsh@harsh-VirtualBox:~$ 

```

Relative or Symbolic mode - In the symbolic mode, we can modify permissions of a specific owner. It makes use of mathematical symbols to modify the file permissions.

```

harsh@harsh-VirtualBox:~$ ls -l cricket.txt
-rwxrwxrwx 1 harsh harsh 29 Aug 25 00:49 cricket.txt
harsh@harsh-VirtualBox:~$ chmod o=rwx cricket.txt
harsh@harsh-VirtualBox:~$ ls -l cricket.txt
-rwxrwxrwx 1 harsh harsh 29 Aug 25 00:49 cricket.txt
harsh@harsh-VirtualBox:~$ 

```

Question 4

Directory permission.

Answer -

Linux permissions dictate 3 things you may do with a file, read, write and execute. They are referred to in Linux by a single letter each.

- r read - you may view the contents of the file.
- w write - you may change the contents of the file.
- x execute - you may execute or run the file if it is a program or script.

For every file or directory, we define 3 sets of people for whom we may specify permissions. owner - a single person who owns the file. (typically the person who created the file but ownership may be granted to someone else by certain users) group - every file belongs to a single group. others - everyone else who is not in the group or is not the owner. Three permissions and three groups of people. That's about all there is to permissions really. Now let's see how we can view and change them.

```
harsh@harsh-VirtualBox:~$ ls -l
total 216
drwxrwxr-x 2 harsh harsh 4096 Aug 31 11:42 ABC
-rwxrwxr-x 1 harsh harsh 16776 Aug 18 14:17 a.out
-rwxrwxr-x 1 harsh harsh 95 Aug 31 00:16 case
-rw-rw-r-- 1 harsh harsh 98 Aug 25 00:40 commands.txt

harsh@harsh-VirtualBox:~$ chmod 000 ABC
harsh@harsh-VirtualBox:~$ ls -l
total 216
d----- 2 harsh harsh 4096 Aug 31 11:42 ABC
-rwxrwxr-x 1 harsh harsh 16776 Aug 18 14:17 a.out
-rwxrwxr-x 1 harsh harsh 95 Aug 31 00:16 case
```

Question 5

Changing file ownership.

Answer -

Every file is owned by a specific user (or UID) and a specific group (or GID). The chown command can be used to change just the user or the user and group of a file. Here is an example of changing the owner of the file test to the user and its group to the user.

```
harsh@harsh-VirtualBox:~$ ls -l cricket.txt
-rwxrwxrwx 1 harsh harsh 29 Aug 25 00:49 cricket.txt
harsh@harsh-VirtualBox:~$ sudo chown root:harsh cricket.txt
[sudo] password for harsh:
harsh@harsh-VirtualBox:~$ ls -l cricket.txt
-rwxrwxrwx 1 root harsh 29 Aug 25 00:49 cricket.txt
```

Question 6

Changing file group.

Answer -

In Linux, each file is associated with an owner and a group and has permissions that determine which users may read, write, or execute the file. Unlike the chown command that allows you to change the user and group ownership, chgrp changes only the group ownership. To find out to which group the file belongs to, use the ls -l command. Regular users can change the group of the file only if they own the file and only to a group of which they are a member.

Administrative users can change the group ownership of all files.

```
harsh@harsh-VirtualBox:~$ ls -l cricket.txt
-rwxrwxrwx 1 root root 29 Aug 25 00:49 cricket.txt
harsh@harsh-VirtualBox:~$ sudo chgrp harsh cricket.txt
harsh@harsh-VirtualBox:~$
```

Assignment 5

Question 1

vi editor and its basics.

Answer -

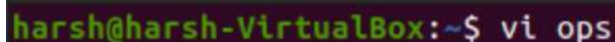
vi is a screen-oriented text editor originally created for the Unix operating system. The default editor that comes with the UNIX operating system is called vi (visual editor). [Alternate editors for UNIX environments include pico and emacs, a product of GNU.] The UNIX vi editor is a full-screen editor and has two modes of operation: Command mode commands which cause the action to be taken on the file, and.

The VI editor is the most popular and classic text editor in the Linux family. Below, are some reasons which make it a widely used editor –

- 1) It is available in almost all Linux Distributions
- 2) It works the same across different platforms and Distributions
- 3) It is user-friendly. Hence, millions of Linux users love it

To open a file in the vi editor to start editing, simply type in 'vi <filename>' in the command prompt. To launch the VI Editor -Open the Terminal (CLI) and type vi <filename_NEW> or <filename_EXISTING> And if you specify an existing file, then the editor would open it for you to edit. Else, you can create a new file.

We can use the vi editor to edit an existing file or to create a new file from scratch. We can also use this editor to just read a text file.



```
harsh@harsh-VirtualBox:~$ vi ops
```



Question 2

Repeat factor.

Answer -

Many vi mode commands may be prefixed by a command repeat factor. This means that most character, change, delete, word, movement, and positioning commands may be preceded by a number which refers to the number of times the command should be repeated.

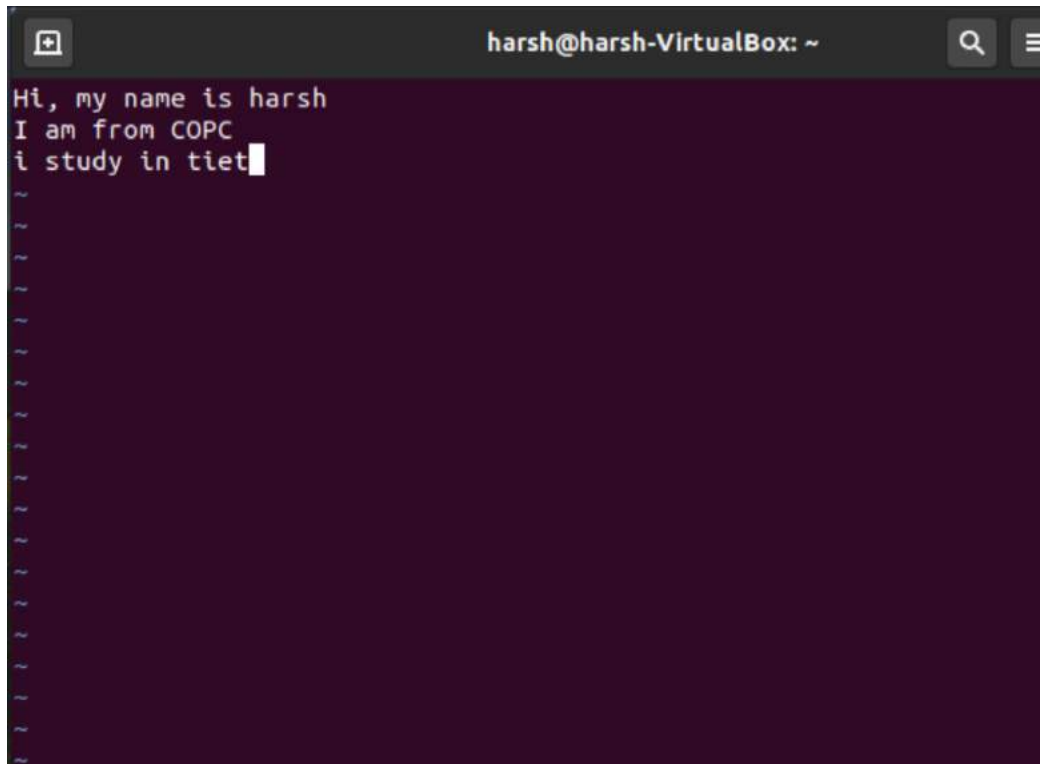
For example, `j` moves the cursor 1 character down, and `4j` moves the cursor 4 characters down. In the commands listed below, examples of using the repeat factor are shown by placing the character `n` in front of a vi mode command. Not all vi mode commands are shown with a repeat factor; just a few, to remind you that it exists!

Question 3

Input mode and insertion of text.

Answer -

vi Editor Insert mode: This mode is for inserting text in the file. You can switch to the Insert mode from the command mode by pressing 'i' on the keyboard. Once you are in Insert mode, any key would be taken as an input for the file on which you are currently working. To return to the command mode and save the changes you have made, you need to press the Esc key.

A screenshot of a terminal window titled 'harsh@harsh-VirtualBox: ~'. The terminal shows the output of a vi editor in insert mode. The text 'Hi, my name is harsh' is on the first line, 'I am from COPC' is on the second line, and 'i study in tiet' is on the third line with a cursor at the end. Below these lines are several lines of tilde (~) characters, indicating the rest of the file or buffer. The terminal has a dark background and a light-colored text.

Question 4

Saving text and quitting.

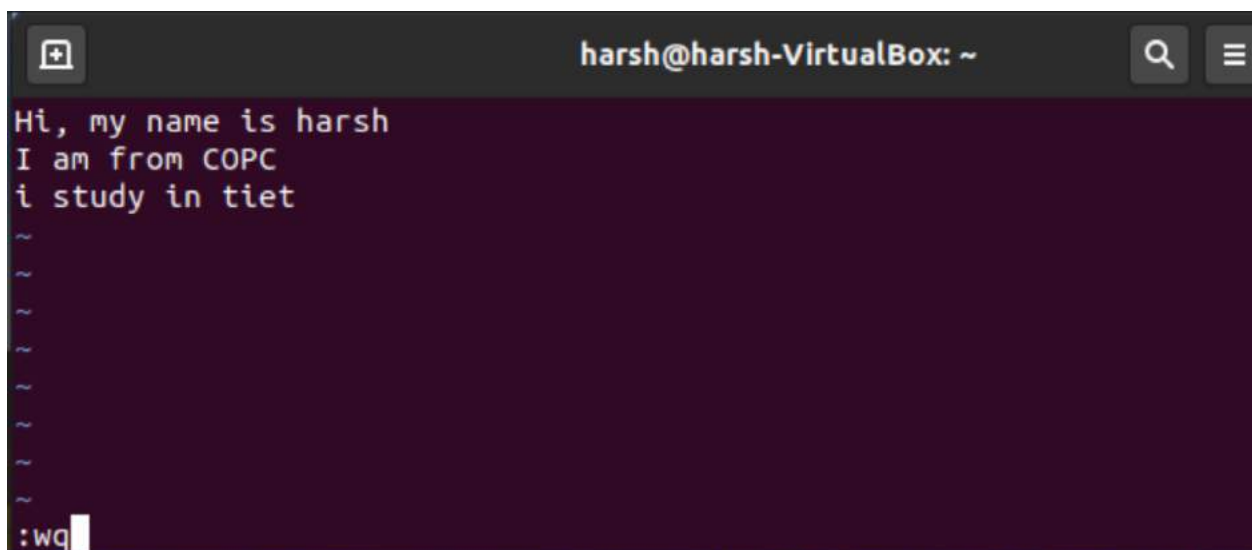
Answer -

The procedure to save a file in vim / vi and quit the editor is as follows:

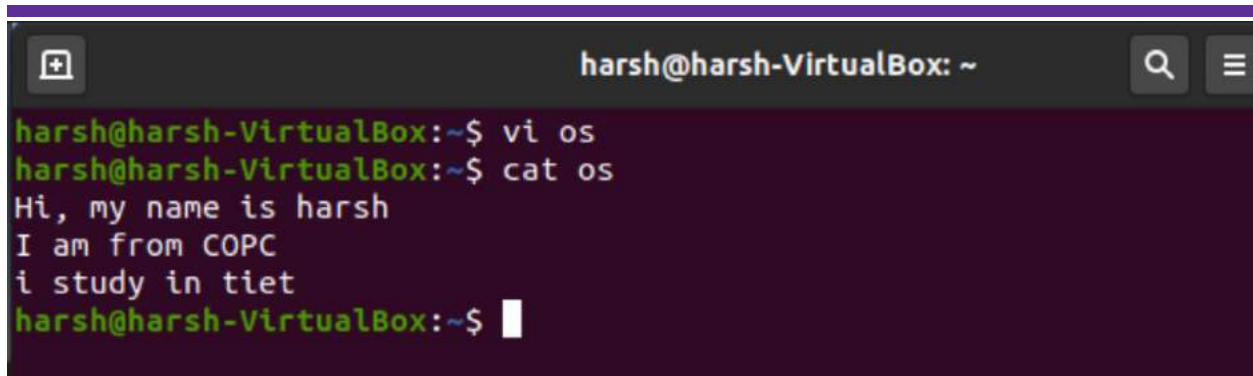
- Open the terminal application in Linux or Unix Next,
- open a file in vim / vi, type: vim filename
- To save a file in Vim / vi, press Esc key, type :w and hit Enter key
- One can save a file and quit vim / Vi by pressing Esc key, type :x and hit Enter key
- ownership may be granted to someone else by certain usersgroup - every file belongs to a single group. others - everyone else who is not in the group or is not the owner.
- Three permissions and three groups of people. That's about all there is to permissions really. Now let's see how we can view and change them.

Saving and Closing the file

- :w - Save the file but keep it open
- :q - Quit without saving
- :wq - Save the file and quit
- :x - saves and quits editing mode



The screenshot shows a terminal window titled "harsh@harsh-VirtualBox: ~". Inside the terminal, the text "Hi, my name is harsh", "I am from COPC", and "i study in tiet" is displayed on the first three lines. Below these lines are several tilde (~) characters. At the bottom left, the command ":wq" is entered, followed by a cursor.



```

harsh@harsh-VirtualBox: ~
harsh@harsh-VirtualBox:~$ vi os
harsh@harsh-VirtualBox:~$ cat os
Hi, my name is harsh
I am from COPC
i study in tiet
harsh@harsh-VirtualBox:~$

```

Question 5

Navigation

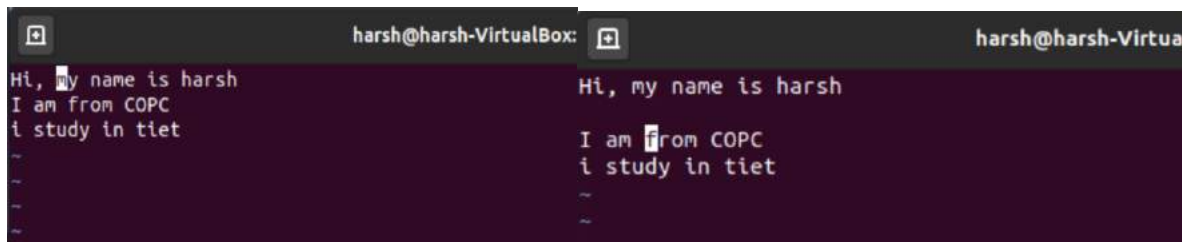
Answer -

If you're using vi from a remote terminal, the arrow keys might not work correctly. The arrow key behaviour depends on your terminal emulator.

If the arrow keys don't work for you, you can use the following substitutes:

- To move left, press h.
 - To move right, press l.
 - To move down, press j.
 - To move up, press k.
-
- ☐ Moving One Word Press w (“word”) to move the cursor to the right one word at a time.
 - ☐ Press b (“back”) to move the cursor to the left one word at a time.
 - ☐ Press W or B to move the cursor past the adjacent punctuation to the next or previous blank space.
 - ☐ Press e (“end”) to move the cursor to the last character of the current word.
 - ☐ Moving to Start or End of Line Press ^ to move the cursor to the start of the current line.
 - ☐ Press \$ to move the cursor to the end of the current line.

- ☐ Moving Down One Line Press the Return key to move the cursor to the beginning of the next line down.
- ☐ Moving Left - Press the BackSpace key to move the cursor one character to the left.
- ☐ Moving Right - Press the Space Bar to move the cursor one character to the right.
- ☐ Moving to the Top - Press H (“high”) to move the cursor to the top of the screen.
- ☐ Moving to the Middle - Press M (“middle”) to move the cursor to the middle of the screen.
- ☐ Moving to the Bottom - Press L (“low”) to move the cursor to the bottom of the screen.



```

harsh@harsh-VirtualBox: ~
Hi, my name is harsh
I am from COPC
i study in tiet
~
~
~

harsh@harsh-VirtualBox: ~
Hi, my name is harsh
I am from COPC
i study in tiet
~
~
~

```

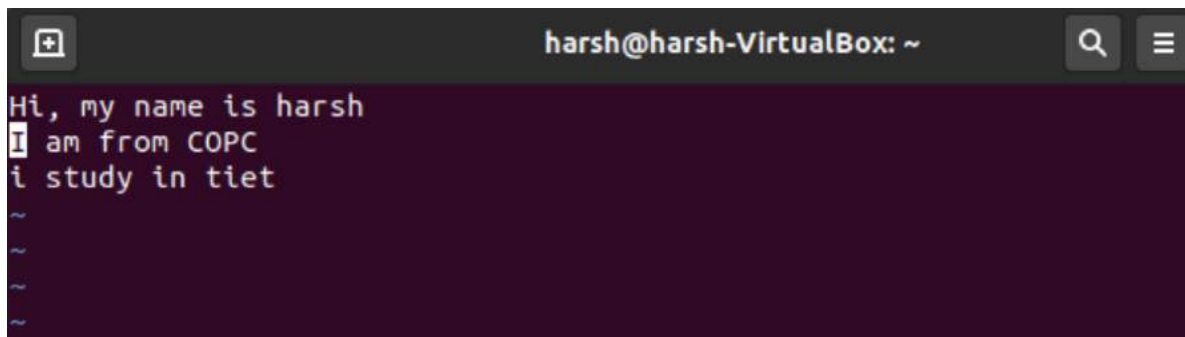
Question 6

Editing text

Answer -

VI Editing commands

- i - Insert at cursor (goes into insert mode)

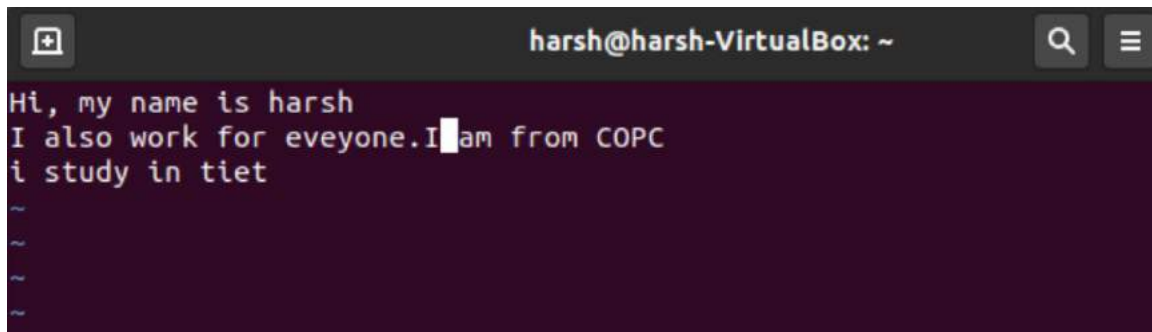


```

harsh@harsh-VirtualBox: ~
Hi, my name is harsh
I am from COPC
i study in tiet
~
~
~

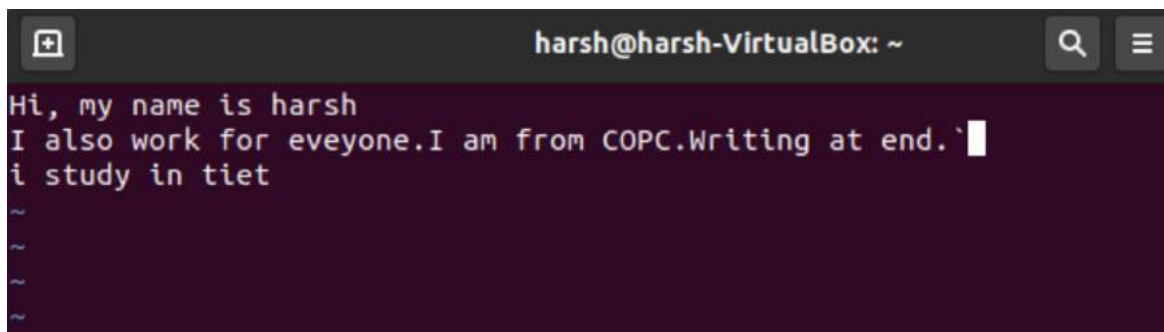
```

- a - Write after cursor (goes into insert mode)



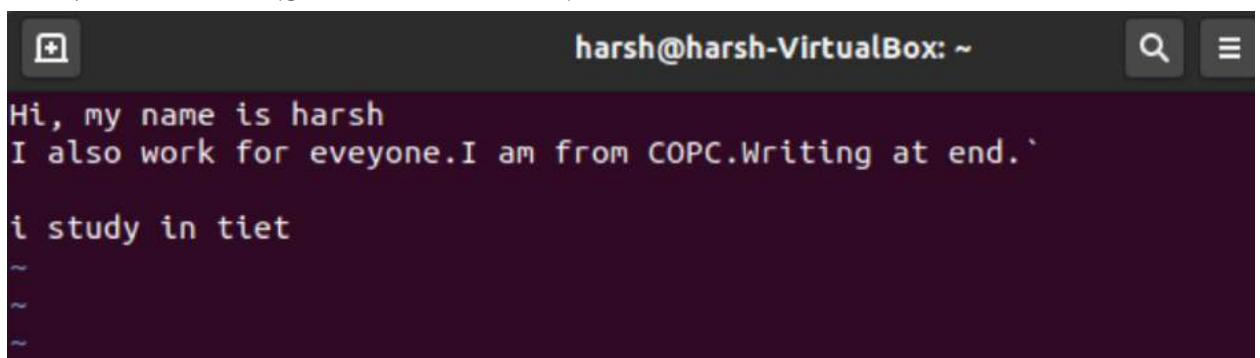
A terminal window titled 'harsh@harsh-VirtualBox: ~' with search and menu icons. The text displayed is:
Hi, my name is harsh
I also work for everyone.I am from COPC
i study in tiet
~
~
~
~

- A - Write at the end of line (goes into insert mode)



A terminal window titled 'harsh@harsh-VirtualBox: ~' with search and menu icons. The text displayed is:
Hi, my name is harsh
I also work for everyone.I am from COPC.Writing at end.`
i study in tiet
~
~
~
~

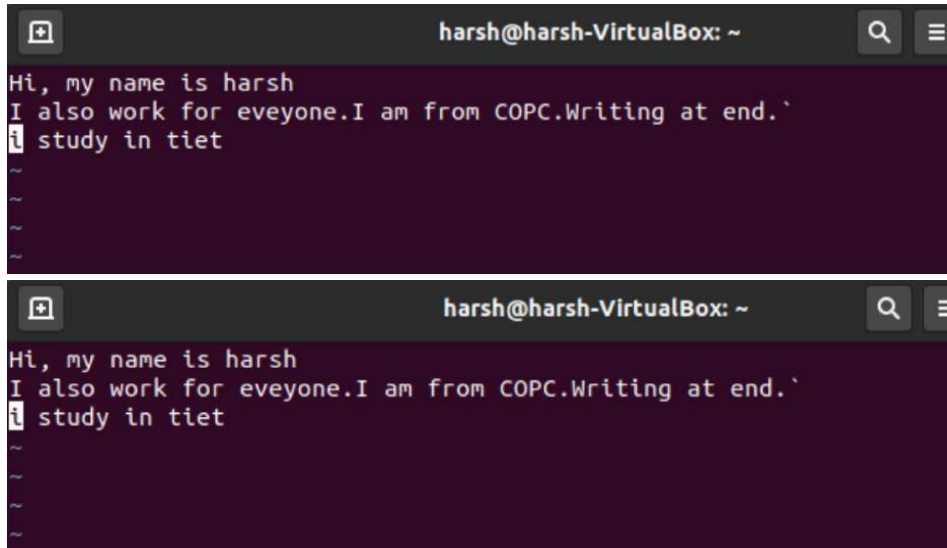
- ESC - Terminate insert mode
- o - Open a new line (goes into insert mode)



A terminal window titled 'harsh@harsh-VirtualBox: ~' with search and menu icons. The text displayed is:
Hi, my name is harsh
I also work for everyone.I am from COPC.Writing at end.`

i study in tiet
~
~
~

- dd - Delete single line

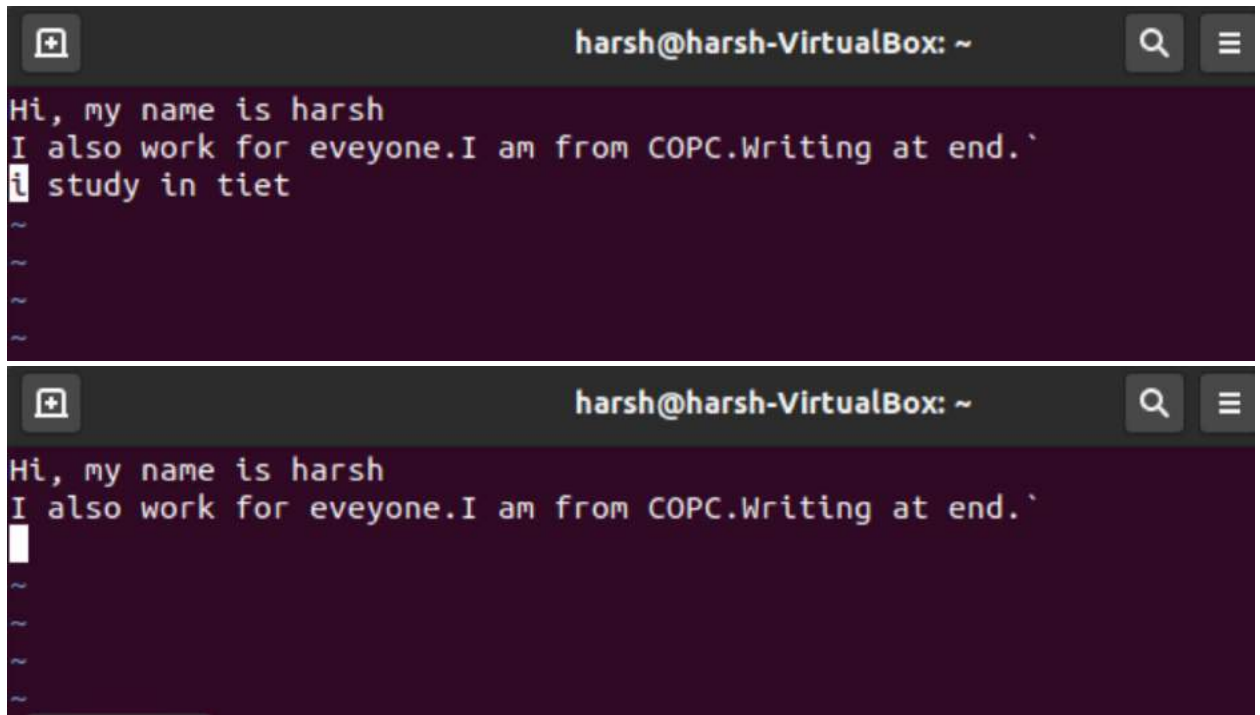


The image shows two terminal windows from a VirtualBox environment. The title bar for both is 'harsh@harsh-VirtualBox: ~'. The top terminal shows the text 'Hi, my name is harsh' followed by 'I also work for everyone.I am from COPC.Writing at end.' on the next line. The cursor is at the start of the third line, which contains 'i study in tiet'. The bottom terminal shows the same text, but the third line is now empty, and the cursor is at the start of a new line.

```
harsh@harsh-VirtualBox: ~  
Hi, my name is harsh  
I also work for everyone.I am from COPC.Writing at end.`  
i study in tiet  
~  
~  
~
```

```
harsh@harsh-VirtualBox: ~  
Hi, my name is harsh  
I also work for everyone.I am from COPC.Writing at end.`  
i study in tiet  
~  
~  
~
```

- D - Delete contents of the line after the cursor

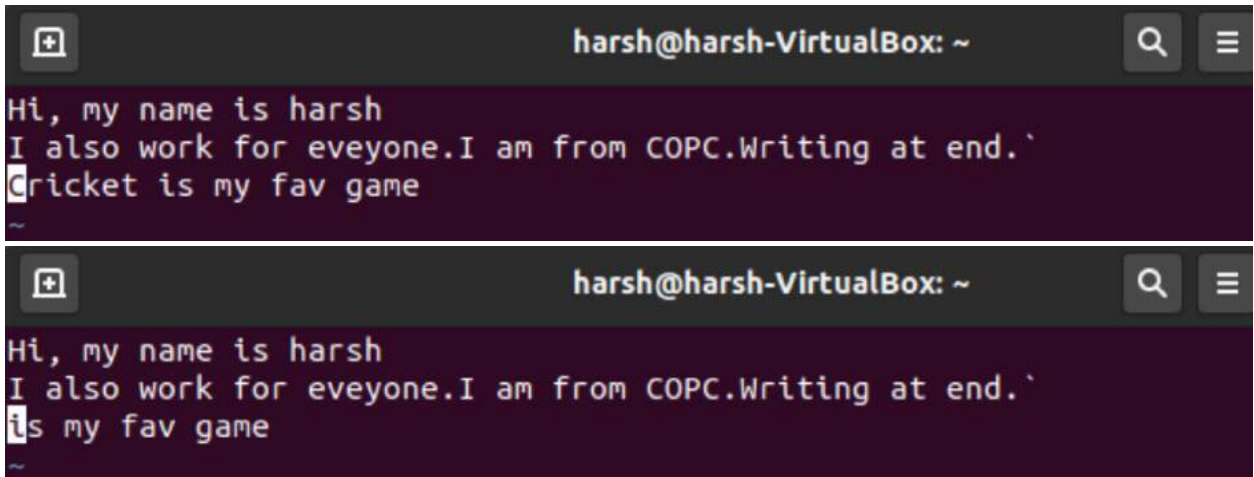


The image shows two terminal windows. The top terminal is the same as the one above, with the cursor at the start of the third line. The bottom terminal shows the result of the 'D' command: the text 'i study in tiet' has been deleted, and the cursor is now at the start of a new line.

```
harsh@harsh-VirtualBox: ~  
Hi, my name is harsh  
I also work for everyone.I am from COPC.Writing at end.`  
i study in tiet  
~  
~  
~
```

```
harsh@harsh-VirtualBox: ~  
Hi, my name is harsh  
I also work for everyone.I am from COPC.Writing at end.`  
~  
~  
~
```

- dw - Delete word



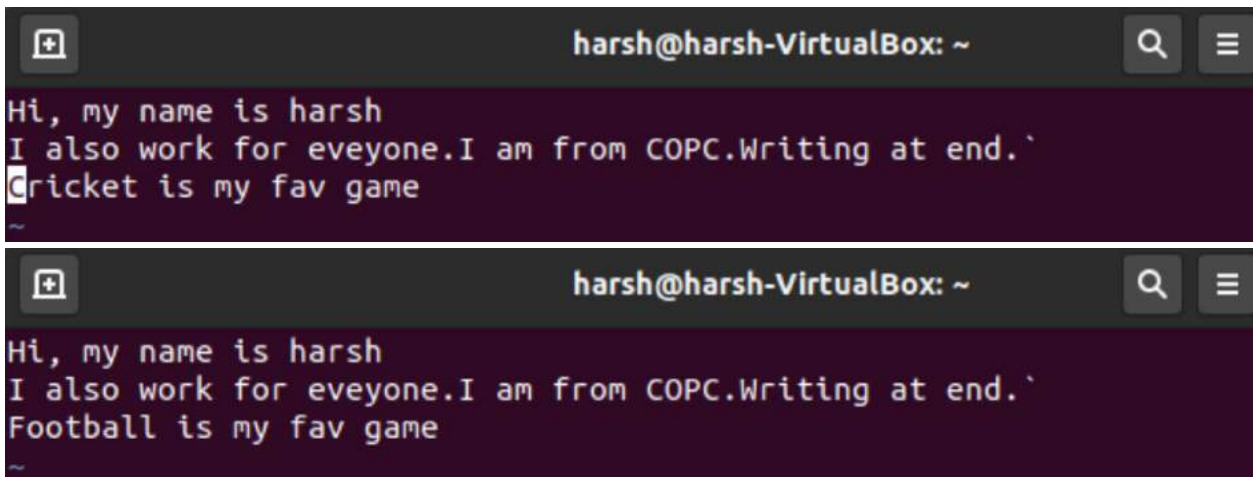
```

harsh@harsh-VirtualBox: ~
Hi, my name is harsh
I also work for everyone.I am from COPC.Writing at end.`
Cricket is my fav game
~

harsh@harsh-VirtualBox: ~
Hi, my name is harsh
I also work for everyone.I am from COPC.Writing at end.`
is my fav game
~

```

- cw - Change word



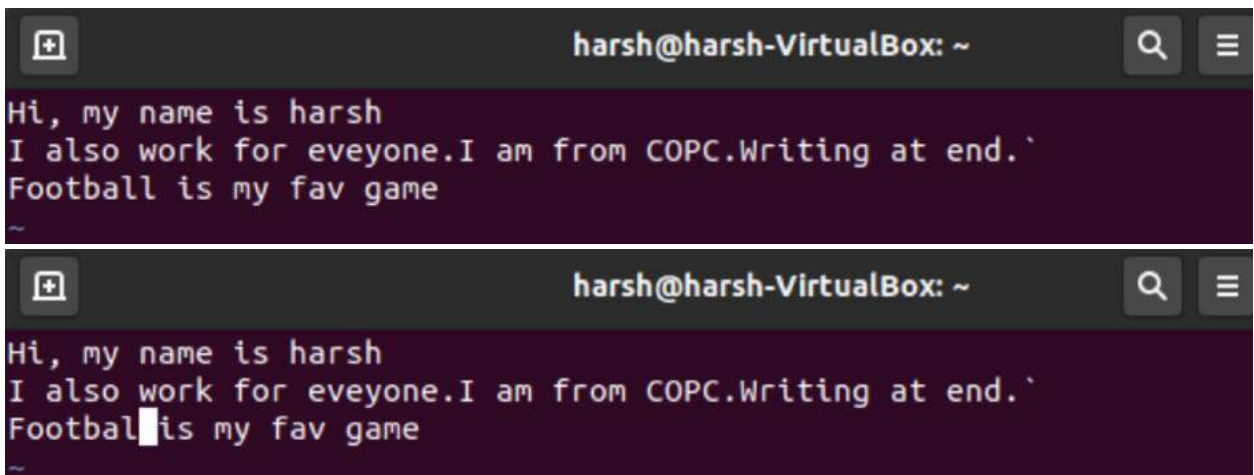
```

harsh@harsh-VirtualBox: ~
Hi, my name is harsh
I also work for everyone.I am from COPC.Writing at end.`
Cricket is my fav game
~

harsh@harsh-VirtualBox: ~
Hi, my name is harsh
I also work for everyone.I am from COPC.Writing at end.`
Football is my fav game
~

```

- x - Delete character at the cursor



```

harsh@harsh-VirtualBox: ~
Hi, my name is harsh
I also work for everyone.I am from COPC.Writing at end.`
Football is my fav game
~

harsh@harsh-VirtualBox: ~
Hi, my name is harsh
I also work for everyone.I am from COPC.Writing at end.`
Footbalis my fav game
~

```

Question 7

Undo the last editing.

Answer -

u - Undo last change

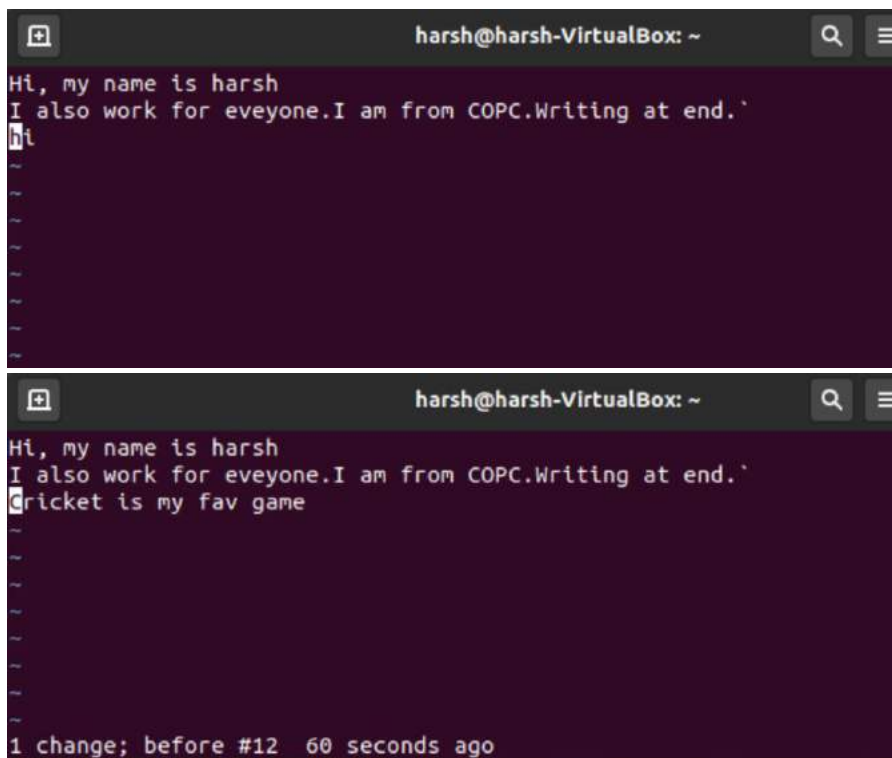
U - Undo all changes to the entire line

In command mode,

- To undo the last change made, we use : 'u'
- To discard all changes made to the current line, we use: 'U'

vim (LINUX) lets us undo and redo multiple editing instructions.

- 'u' behaves differently here; repeated use of this key progressively undoes our previous actions.
- 10u reverses our last 10 editing actions. The function of U remains the same.



The image shows two screenshots of a terminal window titled 'harsh@harsh-VirtualBox: ~'. The first screenshot shows the text 'Hi, my name is harsh' and 'I also work for everyone.I am from COPC.Writing at end.' followed by a cursor on a new line. The second screenshot shows the same text, but the cursor is now on the line 'I also work for everyone.I am from COPC.Writing at end.' and the text 'cricket is my fav game' has been added below it. At the bottom of the second screenshot, it says '1 change; before #12 60 seconds ago'.

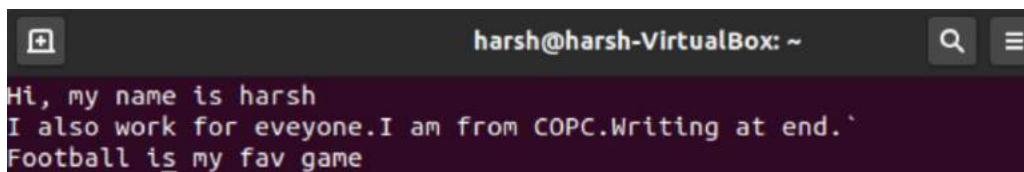
Question 8

Repeating last command.

Answer -

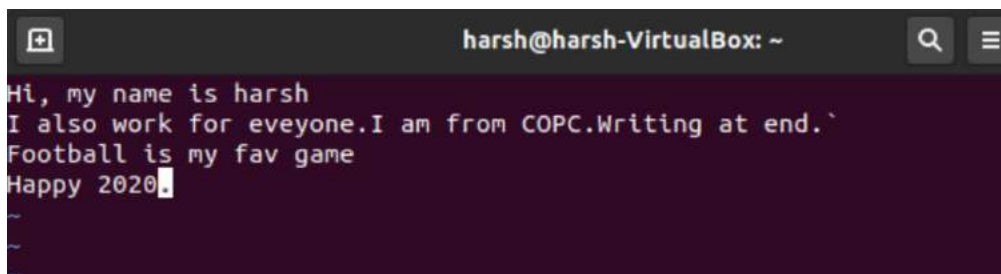
The '.' (dot) command is used for repeating the last instruction in both editing and command mode commands. For example 2dd deletes 2 lines from current line and to repeat this operation, we type '.' (dot)

Before, doing any operation.

A terminal window titled 'harsh@harsh-VirtualBox: ~' with search and menu icons. It contains three lines of text: 'Hi, my name is harsh', 'I also work for everyone.I am from COPC.Writing at end.', and 'Football is my fav game'. The cursor is at the end of the third line.

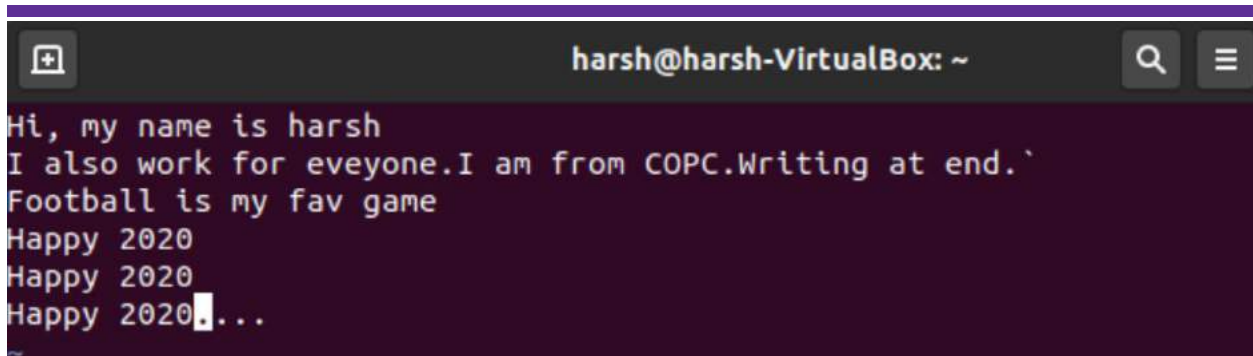
```
harsh@harsh-VirtualBox: ~  
Hi, my name is harsh  
I also work for everyone.I am from COPC.Writing at end.  
Football is my fav game
```

After the last operation. The last operation did is “Happy 2020”

A terminal window titled 'harsh@harsh-VirtualBox: ~' with search and menu icons. It shows the same three lines as the previous screenshot, but with 'Happy 2020' added as a fourth line. The cursor is at the end of the fourth line.

```
harsh@harsh-VirtualBox: ~  
Hi, my name is harsh  
I also work for everyone.I am from COPC.Writing at end.  
Football is my fav game  
Happy 2020
```

Now, every time I use dot(.) command it keeps on adding “Happy 2020”.

A terminal window titled 'harsh@harsh-VirtualBox: ~' with a search icon and a menu icon. The terminal shows a Vim editor session with the following text: 'Hi, my name is harsh', 'I also work for everyone.I am from COPC.Writing at end.', 'Football is my fav game', 'Happy 2020', 'Happy 2020', and 'Happy 2020' followed by a cursor and three dots.

```
harsh@harsh-VirtualBox: ~  
Hi, my name is harsh  
I also work for everyone.I am from COPC.Writing at end.'  
Football is my fav game  
Happy 2020  
Happy 2020  
Happy 2020...
```

Question 9

Searching for a pattern.

Answer -

To search using Vim/vi, for the current word:

- In normal mode, you can search forward or backward.
- One can search forward in vim/vi by pressing / and then typing your search pattern/word.
- To search backward in vi/vim by pressing ? and then typing your search pattern/word.

These are some of the commands used for searching for a pattern.

- / : search forward. ?

```

harsh@harsh-VirtualBox: ~
Hi, my name is harsh
I am from COPC
i study in tiet
~
~
~
~
~
~
~
~
~
~
~
/COPC

```

- ? : search backwards.

```

harsh@harsh-VirtualBox: ~
Hi, my name is harsh
I am from COPC
i study in tiet
~
~
~
~
~
~
~
~
~
~
~
?harsh

```

Question 10

Substitution – search and replace

Answer -

The :substitute command searches for a text pattern, and replaces it with a text string.

There are many options, but these are what you probably want:

- `:%s/pattern_old/new_pattern/g` - Find each occurrence of 'pattern_old' (in all lines), and replace it with 'new_pattern'.

For example, here I have replaced harsh with kashyap.

```

harsh@hHi, my name is kashyap
Hi, my name is harsh      I am from COPC
I am from COPC           i study in tiet
i study in tiet           ~
~                          ~
~                          ~
~                          ~
~                          ~
~                          ~
~                          ~
~                          ~
~                          ~
~                          ~
~                          ~
~                          ~
~                          ~
?harsh                    :%s/harsh/kashyap/g

```

- `:%s/old_pattern/new_pattern/gc` - Change each "old_pattern" to 'new_pattern', but ask for confirmation first.

- 72

